Computational Biology: Genomes, Networks, Evolution
MIT course 6.047/6.878

Taught by Prof. Manolis Kellis

September 25, 2011
Preface and Acknowledgements

These notes summarize the material taught in the MIT course titled “Computational Biology: Genomes, Networks, Evolution”, also cross-listed with Harvard, HST, HSPH and BU over the years. The course was listed as MIT course 6.047/6.878 in 2007-2011 (and under the temporary numbers 6.085/6.095/6.895 in Fall 2005-2006, and 6.096 in Spring 2005). It was cross-listed with MIT/Harvard Health Sciences and Technology (HST) course HST.507 in 2007-2011, Boston University Biological Engineering course BE-562 in 2008 and 2009, and Harvard School of Public Health course IMI231 in 2009-2011.

The course was originally developed by Prof. Manolis Kellis at MIT, with advice from Silvio Micali. It was first taught in Spring 2005 as a half-course extension to the Introduction to Algorithms Course (6.046), and as an independent full-credit course in Fall 2005-2011. The course was co-lectured with Prof. Piotr Indyk in Fall 2005-2006, who contributed to the material on hashing and dimensionality reduction techniques. It was co-taught with Prof. James Galagan in Fall 2007-2009 who contributed to the lectures on expression clustering, supervised learning and metabolic modelling, and who continued teaching the course independently at BU.

The material in the course has benefited tremendously from courses by Bonnie Berger at MIT, whose course “Introduction to Computational Biology (18.417)” was co-taught by Manolis Kellis as a student in Fall 2001, and Serafim Batzoglou at Stanford whose course “Computational Genomics (CS262)” was an inspiration for clarity and style and a source of figures and diagrams for the early chapters on alignment and HMMs. Lastly, the material in the course also benefited from two books used extensively in the course in the last several years, titled “Biological Sequence Analysis” by Durbin, Eddy, Drogh, and Mitchison, and “Bioinformatics Algorithms” by Jones and Pevzner.

The material of several chapters was initially developed by guest lecturers who are experts in their field and contributed new material, figures, slides, organization, and thoughts in the form of one or more lectures. Without them, the corresponding chapters would not have been possible. They are: Pardis Sabeti (Population Genetic Variation), Mark Daly (Medical Genetics), David Reich (Population History), Eric Alm (Bacterial Genomics), John Rinn (Long Non-Coding RNAs), James Galagan (Steady State modeling), Matt Rasmussen (Phylogenomics), Mike Lin (Gene finding), Stefan Washietl (RNA folding), Jason Ernst (Epigenomics), Sushmita Roy (Regulatory Networks), Pouya Kheradpour (Regulatory Genomics).

The Teaching Assistants who taught recitations and help develop the course problem sets have been Reina Reimann (Spring 2005), Pouya Kheradpour (Fall 2005), Matt Rasmussen and Mike Lin (Fall 2006), Mike Lin and David Sontag (Fall 2007), Matt Rasmussen and Pouya Kheradpour (Fall 2008), Ed Reznik and Bob Altshuler (Fall 2009), Matt Edwards (Fall 2010), and Melissa Gymrek (Fall 2011). The notes were originally compiled in a uniform format by Anna Shcherbina (Fall 2011).


The students taking the class and contributing to the scribe notes are:

- Spring 2005: Dan Arlow, Arhab Battacharyya, Punyashloka Biswal, Adam Bouhenguel, Dexter Chan, Shuvo Chatterjee, Tiffany Dobzen, Lyric Doshy, Robert Figueiredo, Edna Gallagher, Josh Grochow, Aleksas Hauser, Blanca Himes, George Hsu, Xiaoming Jia, Scott Johnson, Steven Kannan, Faye Kasemset, Jason Kelly, Daniel Kim, Yuliya Kodysh, Nate Kushman, Lucy Mendel, Jose Pacheco, Sejal Patel, Haiharan Rahul, Gireeja Ranade, Sophie Rapoport, Aditya Rastogi, Shubhangi Saraf, Oded Shalam, Walter Stiehl, Kevin Stolt, James Sun, Xin Sun, Kal Tai, Kah Tai, Chester Tse, Verlik Tzanov, Brian Wu

Taylor, Velin Tzanov, Grant Wang


- Fall 2007: Anton Aboukhalil, Matthew Belmonte, Ellenor Brown, Brad Cater, Alal Eran, Guilherme Fujiwara, Saba Gul, Kate Hoff, Shannon Iyo, Eric Jonas, Peter Kruskall, Michael Lee, Ben Levick, Fulu Li, Alvin Liang, Joshua Lim, Chit-Kwan Lin, Po-Ru Loh, Kevin Modzelewski, Georgis Papachristoudis, Michalis Potamias, Emmanuel Santos, Alex Schwedner, Maryam Shancehi, Timo Somervuo, James Sun, Xin Sun, Robert Toscano, Qingqing Wang, Ning Xie, Qu Zhang, Blaine Ziegler

- Fall 2008: Burak Alver, Tural Badirkhanli, Arnab Bhattacharyya, Can Cenik, Clara Chan, Lydia Chilton, Arkajit Dey, Ardavan Farjadpour, Jeremy Fineman, Bernhard Haeupler, Arman Hajati, Ethan Heilman, Joe Herman, Irwin Jungreis, Arjun Manrai, Nilah Monnier, Christopher Rohde, Rachel Sealfon, Daniel Southern, Paul Steiner, David Stiebel, Mengdi Wang


TODO: missing @scribe: who was this scribed by?

1.1 Introduction and Goals

1.1.1 A course on computational biology

These lecture notes are aimed to be taught as a term course on computational biology, each 1.5 hour lecture covering one chapter, coupled with bi-weekly homework assignments and mentoring sessions to help students accomplish their own independent research projects. The notes grew out of MIT course 6.047/6.878, and very closely reflect the structure of the corresponding lectures.

1.1.2 Duality of Goals: Foundations and Frontiers

There are two goals for this course. The first goal is to introduce you to the foundations of the field of computational biology. Namely, introduce the fundamental biological problems of the field, and learn the algorithmic and machine learning techniques needed for tackling them. This goes beyond just learning how to use the programs and online tools that are popular any given year. Instead, the aim is for you to understand the underlying principles of the most successful techniques that are currently in use, and provide you with the capacity to design and implement the next generation of tools. That is the reason why an introductory algorithms class is set as a pre-req; the best way to gain a deeper understanding for the algorithms presented is to implement them yourself.

The second goal of the course is to tackle the research frontiers of computational biology, and that’s what all the advanced topics and practical assignments are really about. We’d actually like to give you a glimpse of how research works, expose you to current research directions, guide you to find the problems most interesting to you, and help you become an active practitioner in the field. This is achieved through guest lectures, problem sets, labs, and most importantly a term-long independent research project, where you carry out your independent research.

The modules of the course follow that pattern, each consisting of lectures that cover the foundations and the frontiers of each topic. The foundation lectures introduce the classical problems in the field. These problems are very well understood and elegant solutions have already been found; some have even been taught for well over a decade. The frontiers portion of the module cover advanced topics, usually by tackling central questions that still remain open in the field. These chapters frequently include guest lectures by some of the pioneers in each area speaking both about the general state of the field as well as their own lab’s research.

The assignments for the course follow the same foundation/frontiers pattern. Half of the assignments are going to be about working out the methods with pencil on paper, and diving deep into the algorithmic and machine learning notions of the problems. The other half are actually going to be practical questions consisting of programming assignments, where real data sets are provided. You will analyze this data using
the techniques you have learned and interpret your results, giving you a real hands on experience. The assignments build up to the final project, where you will propose and carry out an original research project, and present your findings in conference format. Overall, the assignments are designed to give you the opportunity to apply computational biology methods to real problems in biology.

1.1.3 Duality of disciplines: Computation and Biology

In addition to aiming to cover both foundations and frontiers, the other important duality of this course is between computation and biology.

From the biological perspective of the course, we aim to teach topics that are fundamental to our understanding of biology, medicine, and human health. We therefore shy away from any computationally-interesting problems that are biologically-inspired, but not relevant to biology. We’re not just going to see something in biology, get inspired, and then go off into computer science and do a lot of stuff that biology will never care about. Instead, our goal is to work on problems that can make a significant change in the field of biology. We’d like you to publish papers that actually matter to the biological community and have real biological impact. This goal has therefore guided the selection of topics for the course, and each chapter focuses on a fundamental biological problem.

From the computational perspective of the course, being after all a computer science class, we focus on exploring general techniques and principles that are certainly important in computational biology, but nonetheless can be applied in any other fields that require data analysis and interpretation. Hence, if what you want is to go into cosmology, meteorology, geology, or any such, this class offers computational techniques that will likely become useful when dealing with real-world data sets related to those fields.

1.1.4 Why Computational Biology?

There are many reasons why Computational Biology has emerged as an important discipline in recent years, and perhaps some of these lead you to pick up this book or register for this class. Even though we have our own opinion on what these reasons are, we have asked the students year after year for their own view on what has enabled the field of Computational Biology to expand so rapidly in the last few years. Their responses fall into several broad themes, which we summarize here.

1. Perhaps the most fundamental reason why computational approaches are so well-suited to the study of biological data is that at their core, biological systems are fundamentally digital in nature. To be blunt, humans are not the first to build a digital computer – our ancestors are the first digital computer, as the earliest DNA-based life forms were already storing, copying, and processing digital information encoded in the letters A,C,G, and T. The major evolutionary advantage of a digital medium for storing genetic information is that it can persist across thousands of generations, while analog signals would be diluted from generation to generation from basic chemical diffusion.

2. Besides DNA, many other aspects of biology are digital, such as biological switches, which ensure that only two discrete possible states are achieved by feedback loops and metastable processes, even though these are implemented by levels of molecules. Extensive feedback loops and other diverse regulatory circuits implement discrete decisions through otherwise unstable components, again with design principles similar to engineering practice, making our quest to understand biological systems from an engineering perspective more approachable.

3. Sciences that heavily benefit from data processing, such as Computational Biology, follow a virtuous cycle involving the data available for processing. The more that can be done by processing and analyzing the available data, the more funding will be directed into developing technologies to obtain, process and analyze even more data. New technologies such as sequencing, and high-throughput experimental techniques like microarray, yeast two-hybrid, and ChIP-chip assays are creating enormous and increasing amounts of data that can be analyzed and processed using computational techniques. The $1000 and $100 genome projects are evidence of this cycle. Over ten years ago, when these projects
started, it would have been ludicrous to even imagine processing such massive amounts of data. However, as more potential advantages were devised from the processing of this data, more funding was dedicated into developing technologies that would make these projects feasible.

4. The ability to process data has greatly improved in the recent years, owing to: 1) the massive computational power available today (due to Moore’s law, among other things), and 2) the advances in the algorithmic techniques at hand.

5. Optimization approaches can be used to solve, via computational techniques, that are otherwise intractable problems.

6. **Running time & memory** considerations are critical when dealing with huge datasets. An algorithm that works well on a small genome (for example, a bacteria) might be too time or space inefficient to be applied to 1000 mammalian genomes. Also, combinatorial questions dramatically increase algorithmic complexity.

7. Biological datasets can be **noisy**, and filtering signal from noise is a computational problem.

8. **Machine learning** approaches are useful to make inferences, classify biological features, & identify robust signals.

9. As our understanding of biological systems deepens, we have started to realize that each such system cannot be analyzed in isolation. These systems have proved to be intertwined in ways previously unheard of, and we have started to shift our analyses to techniques that consider them all as a whole.

10. It is possible to use computational approaches to find correlations in an unbiased way, and to come up with conclusions that transform biological knowledge and facilitate active learning. This approach is called **data-driven discovery**.

11. Computational studies can **predict** hypotheses, mechanisms, and theories to explain experimental observations. These falsifiable hypotheses can then be tested experimentally.

12. Computational approaches can be used not only to analyze existing data but also to **motivate data collection** and suggest useful experiments. Also, computational filtering can narrow the experimental search space to allow more focused and efficient experimental designs.

13. Biology has **rules**: Evolution is driven by two simple rules: 1) random mutation, and 2) brutal selection. Biological systems are constrained to these rules, and when analyzing data, we are looking to find and interpret the emerging behavior that these rules generate.

14. **Datasets can be combined** using computational approaches, so that information collected across multiple experiments and using diverse experimental approaches can be brought to bear on questions of interest.

15. Effective **visualizations** of biological data can facilitate discovery.

16. Computational approaches can be used to **simulate & model** biological data.

17. Large scale, systems engineering approaches are facilitated by computational technique to obtain global views into the organism that are too complex to analyze otherwise.

### 1.1.5 Finding Functional Elements: A Computational Biology Question

Several computational biology problems refer to finding biological signals in DNA data (e.g. coding regions, promoters, enhancers, regulators, ...).

We then discussed a specific question that computational biology can be used to address: how can one find functional elements in a genomic sequence? The slide that is filled with letters shows part of the sequence of the yeast genome. Given this sequence, we can ask:
Figure 1.1: In this computational biology problem, we are provided with a sequence of bases, and wish to locate genes and regulatory motifs.

Q: What are the genes that encode proteins?

A: During translation, the start codon marks the first amino acid in a protein, and the stop codon indicates the end of the protein. However, as indicated in the Extracting signal from noise slide, only a few of these ATG sequences in DNA actually mark the start of a gene which will be expressed as protein. The others are noise; for example, they may have been part of introns (non-coding sequences which are spliced out after transcription).

Q: How can we find features (genes, regulatory motifs, and other functional elements) in the genomic sequence?

A: These questions could be addressed either experimentally or computationally. An experimental approach to the problem would be creating a knockout, and seeing if the fitness of the organism is affected. We could also address the question computationally by seeing whether the sequence is conserved across the genomes of multiple species. If the sequence is significantly conserved across evolutionary time, its likely to perform an important function.

There are caveats to both of these approaches. Removing the element may not reveal its function even if there is no apparent difference from the original, this could be simply because the right conditions have not been tested. Also, simply because an element is not conserved doesn't mean it isn't functional. (Also, note that functional element is an ambiguous term. Certainly, there are many types of functional elements in the genome that are not protein-encoding. Intriguingly, 90-95% of the human genome is transcribed (used as a template to make RNA). It isn't known what the function of most of these transcribed regions are, or indeed if they are functional).

1.2 Final Project - Introduction to Research In Computational Biology

lecture1_transcript.html#FinalProject
1.2.1 Final project goals

An important component of being a computational biologist is the ability to carry out independent research in the area. The skills for a successful researcher differ from one person to the next, but in the process of teaching this course, we have identified several aspects that are all needed, and laid out activities for a term-long project, that enable students to carry out their independent research.

The project mirrors real world scientific process: come up with an idea → frame it → propose it → revise it → carry it out → present your results. Students are expected to think critically about their own project, and also evaluate peer research proposals, and lastly respond to feedback from their peers.

Students are expected to use real data and present their results in conference format. The ultimate goal is publishable research. Students are encouraged to talk with the course staff while formulating a final project idea, look ahead through the various chapters and modules, and get an idea of what areas will interest you most.

1.2.2 Final project milestones

Instead of waiting until the end of the term to begin brainstorming or provide feedback, we begin project activities with the first problem set, to identify problems of interest and types of projects, find partners, speak with current students and postdocs in computational biology that can serve as mentors, and lay out a research plan in the style of an NIH proposal to identify potential pitfalls early and address them or work around them before they become a bottleneck.

By setting up several incremental progress milestones throughout the term, coupled with mentoring and feedback throughout the semester, we have achieved consistent progress in previous years, which can be useful to students taking on a new project at any stage of their career. Research projects from this course in the past have been used as the starting point for a published paper, have led to Masters and PhD theses, and earned awards both academically and in conferences.

The timeline for final project is as follows:

1. **Set-up:** a brief overview of your experience and interest. Due 9/26
2. **Brainstorming:** a list of initial project ideas and partners. Due 10/12
3. **Proposal:** submit a project proposal in the form of an NIH proposal. Due 10/24
4. **Review:** review and critique 3 peer proposals. Due 10/31
5. **Midterm Progress Report:** write outline of final report. Due 11/28
6. **Final Project Report:** write report in conference paper format. Due 12/9
7. **Final Class Presentation:** 10min conference talk. Due 12/13

There will be Friday mentoring sessions before each portion of the final project is due, and you are encouraged to find a mentor at the first few sessions who is actively interested in your project and could help you more frequently. The mentoring sessions can be helpful in identifying if unexpected results are the result of a bug or are instead a discovery.

Make sure you start working on the project even while waiting for peer reviews, so that you will have 4-5 weeks to complete the research itself.

1.2.3 Project deliverables

The final project will include the following two deliverables:

1. A written presentation, due Mon at 8pm, last week of classes. The written presentation can contain the following elements:
   - Who did what (to reflect trend in publications)
   - The overall project experience
6.047/6.878 Lecture 01: Introduction and Administrative Details

- Your discoveries
- What you learned from the experience (introspection)

2. An oral presentation, due Thursday after the written presentation. This allows students three days to prepare the oral presentation.

1.2.4 Project grading

Selecting a project that will be successful can be difficult. To help students optimize for a successful project, we let them know in advance the grading scheme, designed to maximize the project impact by being original, challenging, and relevant to the field, but of course the grade is ultimately dependent on the overall achievement and the clarity of presentation.

Briefly, the grading equation for the final project is:

$$\min(O, C, R)xA + P$$

where

Originality - unoriginal computational experiments don’t get published

Challenge - the project needs to be sufficiently difficult

Relevance - it needs to be from biology, can’t just reuse something from another field

Achievement - if you don’t accomplish anything you won’t get a good grade

Presentation - even if you’ve achieved a good project you have to be able to present it so everyone knows that, and make it look easy. The presentation should show how the project is O, C, and R.

Originality, Challenge, Relevance are each out of 5 points, Achievement and Presentation are each out of 10.

1.3 Additional materials

1.3.1 Online Materials for Fall 2011

In addition to these static notes, the course has several online resources:

- The course website can be accessed at: http://compbio.mit.edu/6.047, where course material and information from each term can be found

- A wiki http://6047.wikispaces.com, where students sign up for scribing each lecture

- The course calendar on Google Calendar. You can add ”6.047 Lectures”, a public calendar.

- The NB note-taking system for annotating these notes http://nb.mit.edu/

- Lastly, videos and audio recordings of all lectures from Fall 2010 are freely available online at http://compbio.mit.edu/teaching.html#compbioF10

1.3.2 Textbooks

The following three (optional) reference textbooks recommended for the class.


3. Richard Duda, Peter Hart, David Stork, Pattern Classification.

Each book has a different advantage. The first book is a classic one. It is heavy in math and covers much of what is in class. The book is focused on sequence alignment. As part of sequence alignment theory, the book approaches Hidden Markov Models (HMM), pairwise and multiple alignment methods, phylogenetic trees as well as a short background in probability theory.

The second book intends to balance between mathematical rigorous and biological relevance. According to the author, it is a good book for undergrad students. The book includes a table that associates algorithm to biological problems.

The third book is about machine learning. It uses a more engineering approach. It includes machine learning theory, neural network and, as the name suggests, pattern recognition.

### 1.4 Crash Course in Molecular Biology

For the primarily computational students, we provide a brief introduction to the key notions of molecular biology that we will encounter throughout the term.

#### 1.4.1 The Central Dogma of Molecular Biology

*lecture1_transcript.html#CentralDogma DNA → RNA → Protein*

The central dogma of molecular biology describes how information is stored and used in the cell: The genetic code of an organism is stored in DNA, which is transcribed into RNA, which is then translated into protein, that carries out most processes in the cell.

While the central dogma generally holds true, there are many exceptions to this unidirectional flow of information, such as reverse-transcription from RNA to DNA which is used by retroviruses.

**Did You Know?**

The central dogma is sometimes incorrectly interpreted too strongly as meaning that DNA only stores immutable information from one generation to the next that remains identical within a generation, RNA is only used as a temporary information transfer medium, and proteins are the only molecule that can carry out complex actions.

Again, there are many exceptions to this interpretation, for example:

- Somatic mutations can alter the DNA within a generation, and different cells can have different DNA content.
- Some cells undergo programmed DNA alterations during maturation, resulting in different DNA content, most famously the B and T immunity while blood cells
- Epigenetic modifications of the DNA can be inherited from one generation to the next
- RNA can play many diverse roles in gene regulation, metabolic sensing, and enzymatic reactions, functions that were previously thought to be reserved to proteins.
- Proteins themselves can undergo conformational changes that are epigenetically inherited notably prion states that were famously responsible for mad cow disease

#### 1.4.2 DNA

*DNA → RNA → Protein*

The genetic code of an organism is contained within the DNA molecule. The DNA molecule contains biological signals, some of which encode proteins and some of which encode regulators, signals that turn genes on and off, or decide whether a gene should be turned on or off. Within the genetic code of DNA lies both the data about the proteins that need to be encoded, and the control circuitry, in the form of regulatory motifs.
DNA is the molecular of heredity. DNA is a double helix composed of two complementary strands, with a backbone of phosphate and sugar (deoxyribose) on the outside and bases in the center. Complementary bases are held together by hydrogen bonds. (lecture1_transcript.html#WatsonCrick)

DNA is composed of four nucleotides: A (adenine), C (cytosine), T (thymine), and G (guanine). A and G are purines, which have two rings, while C and T are pyrimidines, with one ring. A and T are connected by two hydrogen bonds, while C and G are connected by three bonds. Therefore, the A-T pairing is weaker than the C-G pairing. (For this reason, the genetic composition of bacteria that live in hot springs is 80% G-C). lecture1_transcript.html#Complementarity

The two DNA strands in the double helix are complementary, meaning that if there is an A on one strand, it will be bonded to a T on the other, and if there is a C on one strand, it will be bonded to a G on the other. The DNA strands also have directionality, which refers to the positions of the pentose ring where the phosphate backbone connects. This directionality convention comes from the fact that DNA and RNA polymerase synthesize in the 5' to 3' direction. With this in mind, we can say that the DNA strands are anti-parallel, as the 5' end of one strand is adjacent to the 3' end of the other. As a result, DNA can be read both in the 3' to 5' direction and the 5' to 3' direction, and genes and other functional elements can be found in each. By convention, DNA is written from 5' to 3'. The 5' and 3' directions refer to the positions on the pentose ring where the phosphate backbone connects.

The structure of DNA, with its weak hydrogen bonds between the bases in the center, allows the strands to easily be separated for the purpose of DNA replication (the capacity for DNA strands to be separated also allows for transcription, translation, recombination, and DNA repair, among others). This was noted by Watson and Crick as It has not escaped our notice that the specific pairing that we have postulated immediately suggests a possible copying mechanism for the genetic material. In the replication of DNA, the two complementary strands are separated, and each of the strands are used as templates for the construction of a new strand.

DNA polymerases attach to each of the strands at the origin of replication, reading each existing strand from the 3 to 5 direction and placing down complementary bases such that the new strand grows in the 5 to 3 direction. Because the new
strand must grow from 5 to 3, one strand (the leading strand) can be copied continuously, while the other (the lagging strand) grows in pieces which are later glued together by DNA ligase. The end result is 2 double-stranded pieces of DNA, where each is composed of 1 old strand, and 1 new strand; for this reason, DNA replication is semiconservative.

DNA needs to be packed tightly, as it is very long, and the cell is small. The DNA molecule is coiled around histone proteins, forming nucleosomes. These are in turn coiled into chromatin fiber. In Eukaryotes, the chromatin is further packaged as chromosomes. The chromosomes are in turn located in the nucleus.

Many organisms have their DNA broken into several chromosomes. Each chromosome contains two strands of DNA, which are complementary to each other but are read in opposite directions. Genes can occur on either strand of DNA. The DNA before a gene (in the 5’ region) is considered “upstream” whereas the DNA after a gene (in the 3’ region) is considered “downstream”.

Before DNA can be replicated or transcribed into RNA, it must be locally unpacked. Thus, gene expression may be regulated by modifications to the chromatin structure, which make it easier or harder for the DNA to be unpacked. This regulation of gene expression via chromatin modification is an example of epigenetics.

### 1.4.3 Transcription

Transcription is the process by which RNA is produced using a DNA template. The DNA is partially unwound to form a bubble, and RNA polymerase attaches to the promoter region of the template strand, reading it from the 3 to 5 direction and placing down complementary bases (except U is used instead of T). The newly produced RNA is grown from the 5 to 3 direction.

In eukaryotes, following transcription, among the modifications that are made to pre-mRNA include splicing out the introns, intervening regions which don’t code for protein, so that only the coding regions, the exons, remain. Different regions of the primary transcript may be spliced out to lead to different protein products (alternative splicing).

### 1.4.4 RNA

RNA is produced when DNA is transcribed. It is structurally similar to DNA, with the following major differences:

1. The nucleotide uracil (U) is used instead of DNA’s thymine (T).
2. RNA contains ribose instead of deoxyribose (deoxyribose lacks the oxygen molecule on the 2 position found in ribose).
3. RNA is single-stranded, whereas DNA is double-stranded.

RNA molecules are the intermediary step to code a protein. RNA molecules also have catalytic and regulatory functions. One example of catalytic function is in protein synthesis, where RNA is part of the ribosome.

There are many different types of RNA, including:

1. **mRNA** (messenger RNA) contains the information to make a protein and is translated into protein sequence.
2. tRNA (transfer RNA) specifies codon-to-amino-acid translation. It contains a 3 base pair anti-codon complementary to a codon on the mRNA, and carries the amino acid corresponding to its anticodon attached to its 3 end.

3. rRNA (ribosomal RNA) forms the core of the ribosome, the organelle responsible for the translation of mRNA to protein.

4. snRNA (small nuclear RNA) is involved in splicing (removing introns from) pre- mRNA, as well as other functions.

Other functional kinds of RNA exist and are still being discovered. RNA molecules can have complex three-dimensional structures and perform diverse functions in the cell.

According to the “RNA world” hypothesis, early life was based entirely on RNA. RNA served as both the information repository (like DNA today) and the functional workhorse (like protein today) in early organisms. Protein is thought to have arisen afterwards via ribosomes, and DNA is thought to have arisen last, via reverse transcription.

1.4.5 Translation

\[ DNA \rightarrow RNA \rightarrow Protein \]

Unlike transcription, in which the nucleotides remained the means of encoding information in both DNA and RNA, when RNA is translated into protein, the primary structure of the protein is determined by the sequence of amino acids of which it is composed. Since there are 20 amino acids and only 4 nucleotides, 3-nucleotides sequences in mRNA, known as codons, encode for each of the 20 amino acids.

Each of the 64 possible 3-sequences of nucleotides (codon) uniquely specifies either a particular amino acid, or is a stop codon that terminates protein translation (the start codon also encodes methionine). Since there are 64 possible codon sequences, the code is degenerate, and some amino acids are specified by multiple encodings. Most of the degeneracy occurs in the 3rd codon position.
### 1.4.6 Protein

DNA → RNA → **Protein**

Protein is the molecule responsible for carrying out most of the tasks of the cell, and can have many functions, such as enzymatic, contractile, transport, immune system, signal and receptor to name a few. Like RNA and DNA, proteins are polymers made from repetitive subunits. Instead of nucleotides, however, proteins are composed of amino acids.

Each amino acid has special properties of size, charge, shape, and acidity. As such, additional structure emerges beyond simply the sequence of amino acids (the primary structure), as a result of interactions between the amino acids. As such, the three-dimensional shape, and thus the function, of a protein is determined by its sequence. However, determining the shape of a protein from its sequence is an unsolved problem in computational biology.

### 1.4.7 Regulation: from Molecules to Life

Not all genes are expressed at the same time in a cell. For example, cells would waste energy if they produced lactose transporter in the absence of lactose. It is important for a cell to know which genes it should expresses and when. A regulatory network is involved to control expression level of genes in a specific circumstance.

Transcription is one of the steps at which protein levels can be regulated. The promoter region, a segment of DNA found upstream (past the 5' end) of genes, functions in transcriptional regulation. The promoter region contains motifs that are recognized by proteins called transcription factors. When bound, transcription factors can recruit RNA polymerase, leading to gene transcription. However, transcription factors can also participate in complex regulatory interactions. There can be multiple binding sites in a promoter, which can act as a logic gate for gene activation. Regulation in eukaryotes can be extremely complex, with gene expression affected not only by the nearby promoter region, but also by distant enhancers and repressors.

We can use probabilistic models to identify genes that are regulated by a given transcription factor. For example, given the set of motifs known to bind a given transcription factor, we can compute the probability...
that a candidate motif also binds the transcription factor (see the notes for precept #1). Comparative sequence analysis can also be used to identify regulatory motifs, since regulatory motifs show characteristic patterns of evolutionary conservation.

The lac operon in E. coli and other bacteria is an example of a simple regulatory circuit. In bacteria, genes with related functions are often located next to each other, controlled by the same regulatory region, and transcribed together; this group of genes is called an operon. The lac operon functions in the metabolism of the sugar lactose, which can be used as an energy source. However, the bacteria prefer to use glucose as an energy source, so if there is glucose present in the environment the bacteria do not want to make the proteins that are encoded by the lac operon. Therefore, transcription of the lac operon is regulated by an elegant circuit in which transcription occurs only if there is lactose but not glucose present in the environment.

![Induction of the lac Operon](image)

Figure 1.6: Operon Lac illustrates a simple biological regulatory system. In the presence of glucose, genes to lactose metabolism are turn out because glucose inactives an activator protein. In the absence of lactose, a repressor protein also turns out the operon. Lactose metabolism genes are expressed only in the presence of lactose and absence of glucose.

### 1.4.8 Modules in Bio-Network

Several genes work together in order to accomplish a specific task. A set of genes which work together is called module. Modules are useful tool to understand the main mechanism responsible to a biological condition. Computational biology tools can be used to find modules as well as use modules and make predictions.

### 1.4.9 Metabolism

Live organisms are made from self-organizing building blocks. Energy source is necessary for organize blocks. The basic mechanism involved in building blocks is degrading small molecules to get energy to build big molecules. The process of degrading molecules to release energy is called catabolism and the process of using energy to assemble more complex molecules is called anabolism. Anabolism and catabolism are both
metabolic processes. Metabolism regulates the flow of mass and energy in order to keep an organism in a state of low entropy.

Enzymes are a critical component of metabolic reactions. The vast majority of (but not all!) enzymes are proteins. Many biologically critical reactions have high activation energies, so that the uncatalyzed reaction would happen extremely slowly or not at all. Enzymes speed up these reactions, so that they can happen at a rate that is sustainable for the cell. In living cells, reactions are organized into metabolic pathways. A reaction may have many steps, with the products of one step serving as the substrate for the next. Also, metabolic reactions often require an investment of energy (notably as a molecule called ATP), and energy released by one reaction may be captured by a later reaction in the pathway. Metabolic pathways are also important for the regulation of metabolic reactions if any step is inhibited, subsequent steps may lack the substrate or the energy that they need to proceed. Often, regulatory checkpoints appear early in metabolic pathways, since if the reaction needs to be stopped, it is obviously better to stop it before much energy has been invested.

1.4.10 Systems Biology

Systems biology strives to explore and explain the behavior that emerges from the complex interactions among the components of a biological system. One interesting recent paper in systems biology is Metabolic gene regulation in a dynamically changing environment (Bennett et al., 2008). This work makes the assumption that yeast is a linear, time invariant system, and runs a signal (glucose) through the system to observe the response. A periodic response to low-frequency fluctuations in glucose level is observed, but there is little response to high-frequency fluctuations in glucose level. Thus, this study finds that yeast acts as a low-pass filter for fluctuations in glucose level.

1.4.11 Synthetic Biology

Not only can we use computational approaches to model and analyze biological data collected from cells, but we can also design cells that implement specific logic circuits to carry out novel functions. The task of designing novel biological systems is known as synthetic biology.

A particularly notable success of synthetic biology is the improvement of artemesinin production. Artemesinin is a drug used to treat malaria. However, artemisinin was quite expensive to produce. Recently, a strain of yeast has been engineered to synthesize a precursor to artemisinic acid at half of the previous cost.

1.4.12 Model organisms and human biology

Diverse model organisms exist for all aspects of human biology. Importance of using model organisms at appropriate level of complexity.

Note: In this particular book, we’ll focus on human biology, and we’ll use examples from baker’s yeast *Saccharomyces cerevisiae*, the fruitfly *Drosophila melanogaster*, the nematode worm *Caenorhabditis elegans*, and the house mouse *Mus musculus*. We’ll deal with bacterial evolution only in the context of metagenomics of the human microbiome.
1.5 Introduction to algorithms and probabilistic inference

1. We will quickly review some basic probability by considering an alternate way to represent motifs: a position weight matrix (PWM). We would like to model the fact that proteins may bind to motifs that are not fully specified. That is, some positions may require a certain nucleotide (e.g. A), while others positions are free to be a subset of the 4 nucleotides (e.g. A or C). A PWM represents the set of all DNA sequences that belong to the motif by using a matrix that stores the probability of finding each of the 4 nucleotides in each position in the motif. For example, consider the following PWM for a motif with length 4:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.6</td>
<td>0.25</td>
<td>0.10</td>
<td>1.0</td>
</tr>
<tr>
<td>G</td>
<td>0.4</td>
<td>0.25</td>
<td>0.10</td>
<td>0.0</td>
</tr>
<tr>
<td>T</td>
<td>0.0</td>
<td>0.25</td>
<td>0.40</td>
<td>0.0</td>
</tr>
<tr>
<td>C</td>
<td>0.0</td>
<td>0.25</td>
<td>0.40</td>
<td>0.0</td>
</tr>
</tbody>
</table>

We say that this motif can generate sequences of length 4. PWMs typically assume that the distribution of one position is not influenced by the base of another position. Notice that each position is associated with a probability distribution over the nucleotides (they sum to 1 and are nonnegative).

2. We can also model the background distribution of nucleotides (the distribution found across the genome):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notice how the probabilities for A and T are the same and the probabilities of G and C are the same. This is a consequence of the complementarity DNA which ensures that the overall composition of A and T, G and C is the same overall in the genome.

3. Consider the sequence $S = \text{GCAA}$.

The probability of the motif generating this sequence is $P(S|M) = 0.4 \times 0.25 \times 0.1 \times 1.0 = 0.01$.

The probability of the background generating this sequence $P(S|B) = 0.4 \times 0.4 \times 0.1 \times 0.1 = 0.0016$.

4. Alone this isn’t particularly interesting. However, given fraction of sequences that are generated by the motif, e.g. $P(M) = 0.1$, and assuming all other sequences are generated by the background ($P(B) = 0.9$) we can compute the probability that the motif generated the sequence using Bayes’ Rule:

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{P(S|M)P(M)}{P(S|B)P(B) + P(S|M)P(M)} = \frac{0.01 \times 0.1}{0.0016 \times 0.9 + 0.01 \times 0.1} = 0.40984$$
1.5.1 Probability distributions
1.5.2 Graphical probabilistic models
1.5.3 Bayes rules: priors, likelihood, posterior
1.5.4 Markov Chains and Sequential Models
1.5.5 Probabilistic inference and learning
1.5.6 Max Likelihood and Max A Posteriori Estimates

Additional materials

Online Materials for Fall 2011

In addition to these static notes, the course has several online resources:

- The course website can be accessed at: http://compbio.mit.edu/6.047, where course material and information from each term can be found.

- A wiki http://6047.wikispaces.com, where students sign up for scribing each lecture.

- The course calendar on Google Calendar. You can add "6.047 Lectures", a public calendar.

- The NB note-taking system for annotating these notes http://nb.mit.edu/

- Lastly, videos and audio recordings of all lectures from Fall 2010 are freely available online at http://compbio.mit.edu/teaching.html#compbioF10

Textbooks

The following three (optional) reference textbooks recommended for the class.


3. Richard Duda, Peter Hart, David Stork, Pattern Classification.

Each book has a different advantage. The first book is a classic one. It is heavy in math and covers much of what is in class. The book is focused on sequence alignment. As part of sequence alignment theory, the book approaches Hidden Markov Models (HMM), pairwise and multiple alignment methods, phylogenetic trees as well as a short background in probability theory.

The second book intends to balance between mathematical rigorous and biological relevance. According to the author, it is a good book for undergrad students. The book includes a table that associates algorithm to biological problems.

The third book is about machine learning. It uses a more engineering approach. It includes machine learning theory, neural network and, as the name sugests, pattern recognition.

Bibliography

[1] lec1test. lec1test, lec1test.
Part I

Comparing Genomes
SEQUENCE ALIGNMENT AND DYNAMIC PROGRAMMING

Guilherme Issao Fujiwara, Pete Kruskal (2007)
Arkajit Dey, Carlos Pards (2008)
Victor Costan, Marten van Dijk (2009)
2.1 Introduction

Evolution has preserved functional elements in the genome. Such preserved elements between species are often homologs\(^1\) – either orthologous or paralogous sequences (refer to Appendix ??). Orthologous gene sequences are of higher interest in the study of evolutionary paths due to the higher influence of purifying selection\(^2\), since such regions are extremely well preserved. A common approach to the analysis of evolutionary similarities is the method of aligning sequences, primarily solved using computational methods (e.g., dynamic programming). These notes discuss the sequence alignment problem, the technique of dynamic programming, and a specific solution of the problem using this technique.

\(^1\)Homologous sequences are genomic sequences descended from a common ancestor.

\(^2\)In the human genome, only \(\approx 5\%\) of the nucleotides are under selection. Appendix ?? has more details on types of selection.
2.2 Aligning Sequences

Sequence alignment represents the method of comparing two or more genetic strands, such as DNA or RNA. These comparisons help with the discovery of genetic commonalities and with the (implicit) tracing of strand evolution. There are two main types of alignment:

- global alignment: an attempt to align every element in a genetic strand, most useful when the genetic strands under consideration are of roughly equal size.\(^3\)

- local alignment: an attempt to align regions of sequences that contain similar sequence motifs within a larger context.

2.2.1 Example Alignment

Within orthologous gene sequences, there are islands of conservation, or relatively large stretches of nucleotides that are preserved between generations. These conserved regions typically imply functional elements and vice versa. Nevertheless, conservation does sometimes occur just by random chance. As an example, we considered the alignment of the Gal10-Gal1 (refer to Figure 2.1) intergenic region for four different yeast species, the first whole genome alignment for crosspieces. As we look at this alignment, we note that some areas are more conserved than others. In particular, we note some small conserved motifs such as CGG and CGC, which in fact are functional elements in the binding of Gal4. This example illustrates how we can read evolution to find functional elements.

![Figure 2.1: Sequence alignment of Gal10-Gal1](image)

\(^3\)Global alignment can also end in gaps.
2.2.2 Solving Sequence Alignment

The genome changes over time (refer to Figure ??), and, lacking a time machine, we cannot compare genomes of living species with their ancestors. Thus, we are limited to comparing just the genomes of living descendants. The goal of sequence alignment is to infer the ‘edit operations’ that change a genome by looking only at these endpoints.

We must make some assumptions when performing sequence alignment, if only because we must transform a biological problem into a computationally feasible one and we require a model with relative simplicity and tractability. In practice, the majority of sequence evolution occurs in the form of nucleotide mutations, deletions, and insertions. Thus, our sequence alignment model will only consider these three operations and will ignore other realistic events that occur with lower probability.

1. A nucleotide mutation is when some nucleotide in a sequence changes to some other nucleotide during the course of evolution.
2. A nucleotide deletion occurs when some nucleotide is deleted from a sequence during the course of evolution.
3. A nucleotide insertion occurs when some nucleotide is added to a sequence during the course of evolution.

Figure 2.2: Evolutionary changes of a genetic sequence

Note that these three events are all reversible. For example, if a nucleotide N mutates into some nucleotide M, it is also possible that nucleotide M can mutate into nucleotide N. Similarly, if nucleotide N is deleted, the event may be reversed if nucleotide N is (re)inserted. Clearly, an insertion event is reversed by a corresponding deletion event.

This reversibility is part of a larger design assumption: time-reversibility. Specifically, any event in our model is reversible in time. For example, a nucleotide deletion going forward in time may be viewed as a nucleotide insertion going backward in time. This is useful because we will be aligning sequences which both exist in the present. In order to compare evolutionary relatedness, we will think of ourselves following one sequence backwards in time to a common ancestor and then continuing forward in time to the other sequence. In doing so, we can avoid the problem of not having an ancestral nucleotide sequence.

Note that time-reversibility is useful in solving some biological problems but does not actually apply to biological systems. For example, CpG (where p denotes the phosphate backbone in a DNA strand) may pair with a TpG or CpA during DNA replication, but the reverse operation cannot occur; it is not time-reversible. To be very clear, time-reversibility is simply a design decision in our model; it is not inherent to the biology.

We also need some way to evaluate our alignments. There are many possible sequences of events that could change one genome into another. Perhaps the most obvious ones minimize the number of events (i.e., mutations, insertions, and deletions) between two genomes, but sequences of events in which many insertions

---

4This is an example where understanding the biology helps the design greatly, and illustrates the general principle that success in computational biology requires strong knowledge of the foundations of both CS and biology. Warning: computer scientists who ignore biology will work too hard.
are followed by corresponding deletions are also possible. We wish to establish an optimality criterion that allows us to pick the ‘best’ series of events describing changes between genomes.

We choose to invoke Occam’s razor and select a maximum parsimony method as our optimality criterion. That is, in general, we wish to minimize the number of events used to explain the differences between two nucleotide sequences. In practice, we find that mutations are more likely to occur than insertions and deletions, and certain mutations are more likely than others. Our parsimony method must take these and other inequalities into account when maximizing parsimony. This leads to the idea of a substitution matrix and a gap penalty, which are developed in the following sections. Note that we did not need to choose a maximum parsimony method for our optimality criterion. We could choose a probabilistic method, for example using Hidden Markov Models (HMMs), that would assign a probability measure over the space of possible event paths and use other methods for evaluating alignments (e.g., Bayesian methods).
2.3 Problem Formulations

In this section, we introduce a simple problem, analyze it, and iteratively increase its complexity until it closely resembles the sequence alignment problem. This section should be viewed as a warm-up for Section ?? on the Needleman-Wunsch algorithm.

2.3.1 Formulation 1: Longest Common Substring

As a first attempt, suppose we treat the nucleotide sequences as strings over the alphabet A, C, G, and T. Given two such strings, S1 and S2, we might try to align them by finding the longest common substring between them. In particular, these substrings cannot have gaps in them.

As an example, if S1 = ACGTCATCA and S2 = TAGTGTCA (refer to Figure ??), the longest common substring between them is GTCA. So in this formulation, we could align S1 and S2 along their longest common substring, GTCA, to get the most matches. A simple algorithm would be to try aligning S1 with different offsets of S2 and keeping track of the longest substring match found thus far. Note that this algorithm is quadratic in the lengths of S1 and S2, which is slower than we would prefer for such a simple problem.

![Figure 2.3: Example of longest common substring formulation](image)

2.3.2 Formulation 2: Longest Common Subsequence (LCS)

Another formulation is to allow gaps in our subsequences and not just limit ourselves to substrings with no gaps. Given a sequence X = (x_1, ..., x_m), we formally define Z = (z_1, ..., z_k) to be a subsequence of X if there exists a strictly increasing sequence i_1 < i_2 < ... < i_k of indices of X such that for all j, 1 ≤ j ≤ k, we have x_{i_j} = z_j (CLRS 350-1).

In the longest common subsequence (LCS) problem, we’re given two sequences X and Y and we want to find the maximum-length common subsequence Z. Consider the example of sequences S1 = ACGTCATCA and S2 = TAGTGTCA (refer to Figure ??). The longest common subsequence is AGTTCA, a longer match than just the longest common substring.
2.3.3 Formulation 3: Sequence Alignment as Edit Distance

The previous LCS formulation is close to the full sequence alignment problem, but so far we have not specified any cost functions that can differentiate between the three types of edit operations (insertion, deletions, and substitutions). Implicitly, our cost function has been uniform, implying that all operations are equally likely. Since substitutions are much more likely, we want to bias our LCS solution with a cost function that prefers substitutions over insertions and deletions.

We recast sequence alignment as a special case of the classic Edit-Distance\(^5\) problem in computer science (CLRS 366). We add varying penalties for different edit operations to reflect biological occurrences. One biological reasoning for this scoring decision is the probabilities of bases being transcribed incorrectly during polymerase. Of the four nucleotide bases, A and G are purines (larger, two fused rings), while C and T are pyrimidines (smaller, one ring). Thus RNA polymerase\(^6\) is much more likely to confuse two purines or two pyrimidines since they are similar in structure. The scoring matrix in Figure ? models the considerations above.

\[
\begin{array}{cccc}
A & G & T & C \\
A & +1 & -\frac{1}{2} & -1 & -1 \\
G & -\frac{1}{2} & +1 & -1 & -1 \\
T & -1 & -1 & +1 & -\frac{1}{2} \\
C & -1 & -1 & -\frac{1}{2} & +1 \\
\end{array}
\]

Figure 2.5: Cost matrix for matches and mismatches

Calculating the scores implies alternating between the probabilistic interpretation of how often biological events occur and the algorithmic interpretation of assigning a score for every operation. The problem is to find the least expensive (as per the cost matrix) operation sequence which can transform the initial nucleotide sequence into the final nucleotide sequence.

---

\(^5\)Edit-distance or Levenshtein distance is a metric for measuring the amount of difference between two sequences (e.g., the Levenshtein distance applied to two strings represents the minimum number of edits necessary for transforming one string into another).

\(^6\)RNA polymerase is an enzyme that helps transcribe a nucleotide sequence.
2.3.4 Formulation 4: Varying Gap Cost Models

Biologically, the cost of creating a gap is not as expensive as the cost of extending an already created gap. Thus, we could create a model that accounts for this cost variation. The principle in the model design is imposing a big initial cost for starting or ending a gap and a smaller incremental cost for each additional character added to an open gap. The gap penalty could also account for disruptions in the coding frame.

2.3.5 Enumeration

It is a known issue that the number of all possible alignments cannot be enumerated (at least when the sequences are lengthy). For example, with two sequences of length 1000, the number of possible alignments exceeds the number of atoms in the universe.

Given a metric to score a given alignment, the simple brute-force algorithm enumerates all possible alignments, computes the score of each one, and picks the alignment with the maximum score. This leads to the question, ‘How many possible alignments are there?’ If you consider only NBAs\(^7\) and \(n > m\), the number of alignments is

\[
\binom{n}{m} = \frac{(n+m)!}{n!m!} \approx \frac{(2n)!}{(n!)^2} \approx \frac{\sqrt{4\pi n (2n)^{2n}}}{e^{2n}} = \frac{2^{2n}}{\sqrt{\pi n}}
\] (2.1)

For some small values of \(n\) such as 100, the number of alignments is already too big (\(> 10^{60}\)) for this enumeration strategy to be feasible. Thus, we using a better algorithm than brute-force is a necessity.

\(^7\)Non-Boring Alignments, or alignments where gaps are always paired with nucleotides.
2.4 Dynamic Programming

Before proceeding to a solution of the sequence alignment problem, we first discuss dynamic programming, a general and powerful method for solving problems with certain types of structure.

2.4.1 Theory of Dynamic Programming

Dynamic programming may be used to solve problems with:

1. **Optimal Substructure**: The optimal solution to an instance of the problem contains optimal solutions to subproblems.

2. **Overlapping Subproblems**: There are a limited number of subproblems, many/most of which are repeated many times.

Dynamic programming is usually, but not always, used to solve optimization problems, similar to greedy algorithms. Unlike greedy algorithms, which require a greedy choice property to be valid, dynamic programming works on a range of problems in which locally optimal choices do not produce globally optimal results. Appendix ?? discusses the distinction between greedy algorithms and dynamic programming in more detail; generally speaking, greedy algorithms solve a smaller class of problems than dynamic programming.

In practice, solving a problem using dynamic programming involves two main parts: Setting up dynamic programming and then performing computation. Setting up dynamic programming usually requires the following 5 steps:

1. Find a 'matrix' parameterization of the problem. Determine the number of dimensions (variables).

2. Ensure the subproblem space is polynomial (not exponential). Note that if a small portion of subproblems are used, then memoization may be better; similarly, if subproblem reuse is not extensive, dynamic programming may not be the best solution for the problem.

3. Determine an effective transversal order. Subproblems must be ready (solved) when they are needed, so computation order matters.

4. Determine a recursive formula: A larger problem is typically solved as a function of its subparts.

5. Remember choices: Typically, the recursive formula involves a minimization or maximization step. Moreover, a representation for storing transversal pointers is often needed, and the representation should be polynomial.

Once dynamic programming is setup, computation is typically straight-forward:

1. Systematically fill in the table of results (and usually traceback pointers) and find an optimal score.

2. Traceback from the optimal score through the pointers to determine an optimal solution.

2.4.2 Fibonacci Numbers

The Fibonacci numbers provide an instructive example of the benefits of dynamic programming. The Fibonacci sequence is recursively defined as $F_0 = F_1 = 1, F_n = F_{n-1} + F_{n-2}$ for $n \leq 2$. We develop an algorithm to compute the $n^{th}$ Fibonacci number, and then refine it first using memoization and later using dynamic programming to illustrate key concepts.

**The Naïve Solution**

The simple top-down approach is to just apply the recursive definition. Listing 1 shows a simple Python implementation.
Figure 2.6: Examples of Fibonacci numbers in nature are ubiquitous

![Fibonacci numbers in nature](image1)

Figure 2.7: The recursion tree for the fib procedure showing repeated subproblems. The size of the tree is $O(\phi^n)$, where $\phi$ is the golden ratio.

![Recursion tree](image2)

Listing 2.1: Python implementation for computing Fibonacci numbers recursively.

```python
def fib(n):
    if n==0 or n==1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

But this top-down algorithm runs in exponential time. That is, if $T(n)$ is how long it takes to compute the $n^{th}$ Fibonacci number, we have that $T(n) = T(n-1) + T(n-2)$, so $T(n) = O(\phi^n)$. The problem is that we are repeating work by solving the same subproblem many times.

The Memoization Solution

A better solution that still utilizes the top-down approach is to memoize the answers to the subproblems. Listing 2 gives a Python implementation that uses memoization.

Listing 2.2: Python implementation for computing Fibonacci numbers using memoization.

```python
from collections import defaultdict

def fib_memo(n):
    memo = defaultdict(int)
    if n==0 or n==1:
        return 1
    if n not in memo:
        memo[n] = fib_memo(n-1) + fib_memo(n-2)
    return memo[n]
```
Assume \( n \) is a non-negative integer.

```python
def fib(n):
    if n not in fibs:
        x = fib(n-2)
        y = fib(n-1)
        fibs[n] = x + y
    return fibs[n]
```

Note that this implementation now runs in \( T(n) = \Theta(n) \) time because each subproblem is computed at most once.

### The Dynamic Programming Solution

For calculating the \( n \)th Fibonacci number, instead of beginning with \( F(n) \) and using recursion, we can start computation from the bottom since we know we are going to need all of the subproblems anyway. In this way, we will omit much of the repeated work that would be done by the naive top-down approach, and we will be able to compute the \( n \)th Fibonacci number in \( \Theta(n) \) time. Listing 3 shows a Python implementation of this approach.

Listing 2.3: Python implementation for computing Fibonacci numbers iteratively using dynamic programming.

```python
Assume \( n \) is a non-negative integer
def fib(n):
    x = y = 1
    for i in range(1, n):
        x, y = y, x + y
    return y
```

This method is optimized to only use constant space instead of an entire table since we only need the answer to each subproblem once. But in general dynamic programming solutions, we want to store the solutions to subproblems in a table since we may need to use them multiple times without recomputing their answers. Such solutions would look somewhat like the memoization solution in Listing 2, but they will generally be bottom-up instead of top-down. In this particular example, the distinction between the memoization solution and the dynamic programming solution is minimal as both approaches compute all subproblem solutions and use them the same number of times. In general, memoization is useful when not all subproblems will be computed, while dynamic programming may not have as much overhead as memoization when all subproblem solutions must be calculated. Additional dynamic programming examples may be found at [http://people.csail.mit.edu/bdean/6.046/dp/](http://people.csail.mit.edu/bdean/6.046/dp/).

### 2.4.3 Sequence Alignment using Dynamic Programming

We are now ready to solve the more difficult problem of sequence alignment using dynamic programming, which is presented in depth in the next section. Note that the key insight in solving the sequence alignment problem is that alignment scores are additive. This allows us to create a matrix \( M \) indexed by \( i \) and \( j \), which are positions in two sequences \( S \) and \( T \) to be aligned. The best alignment of \( S \) and \( T \) corresponds with the best path through the matrix \( M \) after it is filled in using a recursive formula.

By using dynamic programming to solve the sequence alignment problem, we achieve a provably optimal solution that is far more tractable than brute-force enumeration.
2.5 The Needleman-Wunsch Algorithm

We will now use dynamic programming to tackle the harder problem of general sequence alignment. Given two strings \( S = (S_1, \ldots, S_n) \) and \( T = (T_1, \ldots, T_m) \), we want to find the longest common subsequence, which may or may not contain gaps. Rather than maximizing the length of a common subsequence we want to compute the common subsequence that optimizes the score as defined by our scoring function. Let \( d \) denote the gap penalty cost and \( s(x; y) \) the score of aligning a base \( x \) and a base \( y \). These are inferred from insertion/deletion and substitution probabilities which can be determined experimentally or by looking at sequences that we know are closely related. The algorithm we will develop in the following sections to solve sequence alignment is known as the Needleman-Wunsch algorithm.

2.5.1 Problem Statement

Suppose we have an optimal alignment for two sequences \( S \) and \( T \) in which \( S_i \) matches \( T_j \). The key insight is that this optimal alignment is composed of an optimal alignment between \( (S_1, \ldots, S_{i+1}) \) and \( (T_1, \ldots, T_{j+1}) \) and an optimal alignment between \( (S_{i+1}, \ldots, S_n) \) and \( (T_{j+1}, \ldots, T_m) \). This follows from a cut-and-paste argument: if one of the alignments is suboptimal, then we cut-and-paste a better alignment in place of the suboptimal one, which achieves a higher score of the overall alignment contradicting the optimality of the overall alignment. Notice that the score is additive, so the score of the overall alignment equals the addition of the scores of the alignments of the subsequences. This implicitly assumes that the sub-problems of computing the optimal scoring alignments of the subsequences are independent. We need to biologically motivate that such an assumption leads to meaningful results.

2.5.2 Top-down and bottom-up approach

In a top-down recursive approach we can use memoization to create a potentially large dictionary indexed by each of the subproblems that we are solving (aligned sequences). This needs \( O(n^2 m^2) \) space if we index each subproblem by the starting and end points of the subsequences for which an optimal alignment needs to be computed. The advantage is that we solve each subproblem at most once: if it is not in the dictionary, the problem gets computed and then inserted into dictionary for further reference.

In a bottom-up iterative approach we can use dynamic programming. We define the order of computing sub-problems in such a way that a solution to a problem is computed once the relevant sub-problems have been solved. In particular, simpler sub-problems will come before more complex ones. This removes the need for keeping track of which sub-problems have been solved (the dictionary in memoization turns into a matrix) and ensures that there is no duplicated work (each sub-alignment is computed only once).

2.5.3 Index space of ubproblems

We need to index the space of subproblems. Let \( F_{i,j} \) be the score of the optimal alignment of \( (S_1, \ldots, S_i) \) and \( (T_1, \ldots, T_j) \). This allows us to maintain an \( m \times n \) matrix \( F \) with the solutions (i.e. optimal scores) for all the subproblems.

2.5.4 Local optimality

We can compute the optimal solution for a subproblem by making a locally optimal choice based on the results from the smaller sub-problems. Thus, we need to establish a recursive function that shows how the solution to a given problem depends on its subproblems. And we use this recursive definition to fill up the table \( F \) in a bottom-up fashion.

We can consider the 4 possibilities (insert, delete, substitute, match) and evaluate each of them based on the results we have computed for smaller subproblems. To initialize the table, we set \( F_{0,j} = -j \cdot d \) and \( F_{i,0} = -i \cdot d \) since those are the scores of aligning \( (T_1, \ldots, T_j) \) with \( j \) gaps and \( (S_1, \ldots, S_i) \) with \( i \) gaps. Then we traverse the matrix column by column computing the optimal score for each alignment subproblem by considering the four possibilities:

- Sequence \( S \) has a gap at the current alignment position.
• Sequence T has a gap at the current alignment position.
• There is a mutation (nucleotide substitution) at the current position.
• There is a match at the current position.

We then use the possibility that produces the maximum score. We express this mathematically by the recursive formula for $F_{i,j}$

$$F_{i,j} = \max \begin{cases} 
F_{i,j-1} - d & \text{insert gap in } T \\
F_{i-1,j} - d & \text{insert gap in } S \\
F_{i-1,j-1} + s(S_i; T_j) & \text{match or mutation}
\end{cases}$$

(2.2)

After traversing the matrix, the optimal score for the global alignment is given by $F_{m,n}$. The traversal order needs to be such that we have solutions to given subproblems when we need them. Namely, to compute $F_{i,j}$, we need to know the values to the left, up, and diagonally above $F_{i,j}$ in the table. Thus we can traverse the table in row or column major order or even diagonally from the top left cell to the bottom right cell. Now, to obtain the actual alignment we just have to remember the choices that we made at each step.

2.5.5 Optimal Solution

Note the duality between paths through the matrix F and sequence alignments. In evaluating each cell $F_{i,j}$ we make a choice by selecting the maximum of the three possibilities. Thus the value of each (uninitialized) cell in the matrix is determined either by the cell to its left, above it, or diagonally to the left above it. A match and a substitution are both represented as traveling in the diagonal direction; however, a different cost can be applied for each, depending on whether the two base pairs we are aligning match or not. To construct the actual optimal alignment, we need to traceback through our choices in the matrix. It is helpful to maintain a pointer for each cell while filling up the table that shows which choice was made to get the score for that cell. Then we can just follow our pointers backwards to reconstruct the optimal alignment.

2.5.6 Solution Analysis

The runtime analysis of this algorithm is very simple. Each update takes $O(1)$ time, and since there are $mn$ elements in the matrix F, the total running time is $O(mn)$. Similarly, the total storage space is $O(mn)$. For the more general case where the update rule is more complicated, the running time may be more expensive. For instance, if the update rule requires testing all sizes of gaps (e.g. the cost of a gap is not linear), then the running time would be $O(mn(m + n))$.

2.5.7 Needleman-Wunsch in practice

Assume we want to align two sequences S and T\(^8\), where

$S = \text{ABCJRQCLCRPM}$

$T = \text{AJCJNRCKCRBP}$

The first step is placing the two sequences along the margins of a matrix and also initialize the matrix cells with a 1 anywhere two sequences match and 0 otherwise, as in Figure ?? below.

To find the best route one performs two passes through the matrix. The first pass traces a score for all possible routes and moves right to left, bottom to top, as in Figure ?? . Once the score for all possible routes are found, the maximum can be chosen (it will be somewhere on the topmost row or leftmost column) and a second pass can be carried out, this time running left to right, top to bottom to find that alignment that gives the maximum score.

Proceeding right to left and bottom to top and left, each element of the matrix gets added to the largest number from the row or column to the lower right.

The final optimal alignment is traced from left to right of the matrix, top to bottom choosing the largest number available in each step. Multiple optimal pathways are shown , as in Figure ?? . A pseudocode implementation of the Needleman-Wunsch algorithm is included in Appendix ??

---

\(^8\)Example taken from [http://helix.mcmaster.ca/721/outline2/node39.html](http://helix.mcmaster.ca/721/outline2/node39.html)
### Figure 2.8: (Example) Initial setup for Needleman-Wunsch

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>N</th>
<th>J</th>
<th>R</th>
<th>Q</th>
<th>C</th>
<th>L</th>
<th>C</th>
<th>R</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Figure 2.9: (Example) Half-way through the second step of Needleman-Wunsch

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>N</th>
<th>J</th>
<th>R</th>
<th>Q</th>
<th>C</th>
<th>L</th>
<th>C</th>
<th>R</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>K</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Figure 2.10: (Example) Tracing the optimal alignment

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>N</th>
<th>J</th>
<th>R</th>
<th>Q</th>
<th>C</th>
<th>L</th>
<th>C</th>
<th>R</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>K</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
2.6 Extensions and optimizations of the basic model

The algorithm we presented is much faster than the brute-force strategy of enumerating alignments, and it performs well for sequences up to 10 kilo-bases long. However, for whole genome alignments, the algorithm given is not feasible, but we can make modifications to it to further improve its performance.

2.6.1 Bounded Dynamic Programming

One possible optimization is to ignore Mildly Boring Alignments (MBAs), or alignments that have too many gaps. Explicitly, can limit ourselves to stay within some distance $W$ from the diagonal in the matrix $F$ of subproblems. That is, we assume that the optimizing path in $F$ from $F_{0,0}$ to $F_{m,n}$ is within distance $W$ along the diagonal. This means that recursion (1) only needs to be applied to the entries in $F$ within distance $W$ around the diagonal, and this yields a time/space cost of $O((m + n)W)$ (refer to Figure 2.11).

Figure 2.11: Bounded dynamic programming example

Note, however, that this strategy is heuristic and no longer guarantees an optimal alignment. Instead it attains a lower bound on the optimal score. This can be used in a subsequent step where we discard the recursions in matrix $F$ which, given the lower bound, cannot lead to an optimal alignment.

2.6.2 Linear Space Alignment

Recursion (1) can be solved using only linear space: we update the columns in $F$ from left to right during which we only keep track of the last updated column which costs $O(m)$ space. However, besides the score $F_{m,n}$ of the optimal alignment, we also want to compute a corresponding alignment. If we use trace back, then we need to store pointers for each of the entries in $F$, and this costs $O(mn)$ space.

It is possible to also find an optimal alignment using only linear space! The idea is to use divide and conquer in order to compute the structure of the optimal alignment for one entry in each step. Figure 2.12 illustrates the process. We first compute the row index $u \in \{1, \ldots, m\}$ that is on the optimal path while crossing the $\frac{n}{2}$-th column. For $1 \leq i \leq m$ and $\frac{n}{2} \leq j \leq n$ let $C_{i,j}$ denote the row index that is on the optimal path to $F_{i,j}$ while crossing the $\frac{n}{2}$-th column. Then, while we update the columns of $F$ from left to right, we can also update the columns of $C$ from left to right. So, in $O(mn)$ time and $O(m)$ space we are able to compute the score $F_{m,n}$ and also $C_{m,n}$, which is equal to the row index $u \in \{1, \ldots, m\}$ that is on the optimal path while crossing the $\frac{n}{2}$-th column.
Now the idea of divide and conquer kicks in. We repeat the above procedure for the upper left $u \times \frac{n}{2}$ submatrix of $F$ and also repeat the above procedure for the lower right $(m - u) \times \frac{n}{2}$ submatrix of $F$. This can be done using $O(m + n)$ allocated linear space. The running time for the upper left submatrix is $O\left(\frac{un}{2}\right)$ and the running time for the lower right submatrix is $O\left(\frac{(m-u)n}{2}\right)$, which added together gives a running time of $O\left(\frac{mn}{2}\right) = O(mn)$.

We keep on repeating the above procedure for smaller and smaller submatrices of $F$ while we gather more and more entries of an alignment with optimal score. The total running time is $O(mn) + O\left(\frac{mn}{2}\right) + O\left(\frac{mn}{4}\right) + ... = O(2mn) = O(mn)$. So, without sacrificing the overall running time (up to a constant factor), divide and conquer leads to a linear space solution!
2.7 Multiple sequence alignment
2.7.1 Pairwise-to-multiple alignment heuristics
2.7.2 Progressive alignment
2.7.3 Joint alignment and phylogeny problem
2.8 Current Research Directions
2.9 Further Reading
2.10 Tools and Techniques
2.11 What Have We Learned?
2.12 Appendix

2.12.1 Homology
One of the key goals of sequence alignment is to identify homologous sequences (e.g., genes) in a genome. Two sequences that are homologous are evolutionarily related, specifically by descension from a common ancestor. The two primary types of homologs are orthologus and paralogus (refer to Figure ??). Other forms of homology exist (e.g., xenologs), but they are outside the scope of these notes.

Orthologs arise from speciation events, leading to two organisms with a copy of the same gene. For example, when a single species A speciates into two species B and C, there are genes in species B and C that descend from a common gene in species A, and these genes in B and C are orthologous (the gene evolve independently, but still perform the same relative function).

Paralogs arise from duplication events within a species. For example, when a gene duplication occurs in some species A, the species has an original gene B and a gene copy B’, and the genes B and B’ are paralogus.

Generally, orthologous sequences between two species will be more closely related to each other than paralogous sequences. This occurs because orthologous typically (although not always) preserve function over time, whereas paralogous often change over time, for example by specializing a gene’s (sub)function or by evolving a new function. As a result, determining orthologous sequences is generally more important than identifying paralogous sequences when gauging evolutionary relatedness.

2.12.2 Natural Selection
The topic of natural selection is a too large topic to summarize effectively in just a few short paragraphs; instead, this appendix introduces three broad types of natural selection: positive selection, negative selection, and neutral selection.

- **Positive selection** occurs when a trait is evolutionarily advantageous and increases an individual’s fitness, so that an individual with the trait is more likely to have (robust) offspring. It is often associated with the development of new traits.

- **Negative selection** occurs when a trait is evolutionarily disadvantageous and decreases an individual’s fitness. Negative selection acts to reduce the prevalence of genetic alleles that reduce a species’ fitness. Negative selection is also known as purifying selection due to its tendency to ‘purify’ genetic alleles until only a single allele exists in the population.

- **Neutral selection** describes evolution that occurs randomly, as a result of alleles not affecting an individual’s fitness. In the absence of selective pressures, no positive or negative selection occurs, and the result is neutral selection.

9R.B. - BIOS 60579
2.12.3 Dynamic Programming v. Greedy Algorithms

Dynamic programming and greedy algorithms are somewhat similar, and it behooves one to know the distinctions between the two. Problems that may be solved using dynamic programming are typically optimization problems that exhibit two traits:

1. **optimal substructure** and
2. **overlapping subproblems**.

Problems solvable by greedy algorithms require both these traits as well as (3) the **greedy choice property**. When dealing with a problem “in the wild,” it is often easy to determine whether it satisfies (1) and (2) but difficult to determine whether it must have the greedy choice property. It is not always clear whether locally optimal choices will yield a globally optimal solution.

For computational biologists, there are two useful points to note concerning whether to employ dynamic programming or greedy programming. First, if a problem may be solved using a greedy algorithm, then it may be solved using dynamic programming, while the converse is not true. Second, the problem structures that allow for greedy algorithms typically do not appear in computational biology.

To elucidate this second point, it could be useful to consider the structures that allow greedy programming to work, but such a discussion would take us too far afield. The interested student (preferably one with a mathematical background) should look at matroids and greedoids\(^\text{10}\), which are structures that have the greedy choice property. For our purposes, we will simply state that biological problems typically involve entities that are highly systemic and that there is little reason to suspect sufficient structure in most problems to employ greedy algorithms.

2.12.4 Pseudocode for the Needleman-Wunsch Algorithm

The first problem in the first problem set asks you to finish an implementation of the Needleman-Wunsch (NW) algorithm, and working Python code for the algorithm is intentionally omitted. Instead, this appendix summarizes the general steps of the NW algorithm (Section ??) in a single place.

Problem: Given two sequences $S$ and $T$ of length $m$ and $n$, a substitution matrix $vU$ of matching scores, and a gap penalty $G$, determine the optimal alignment of $S$ and $T$ and the score of the alignment.

Algorithm:

1. Create two $(m + 1) \times (n + 1)$ matrices $A$ and $B$. $A$ will be the scoring matrix, and $B$ will be the traceback matrix. The entry $(i, j)$ of matrix $A$ will hold the score of the optimal alignment of the sequences $S[1, \ldots, i]$ and $T[1, \ldots, j]$, and the entry $(i, j)$ of matrix $B$ will hold a pointer to the entry from which the optimal alignment was built.

2. Initialize the first row and column of the score matrix $A$ such that the scores account for gap penalties, and initialize the first row and column of the traceback matrix $B$ in the obvious way.

3. Go through the entries $(i, j)$ of matrix $A$ in some reasonable order, determining the optimal alignment of the sequences $S[1, \ldots, i]$ and $T[1, \ldots, j]$ using the entries $(i - 1, j - 1)$, $(i - 1, j)$, and $(i, j - 1)$. Set the pointer in the matrix $B$ to the corresponding entry from which the optimal alignment at $(i, j)$ was built.

4. Once all entries of matrices $A$ and $B$ are completed, the score of the optimal alignment may be found in entry $(m, n)$ of matrix $A$.

5. Construct the optimal alignment by following the path of pointers starting at entry $(m, n)$ of matrix $B$ and ending at entry $(0, 0)$ of matrix $B$.

Bibliography
CHAPTER THREE

RAPID SEQUENCE ALIGNMENT AND DATABASE SEARCH

Maria Rodriguez (Sep 13, 2011)
Rushil Goel (Sep 16, 2010)
Eric Eisner - Guilhelm Richard (Sep 17, 2009)
Tural Badirkhnali (Sep 11, 2008)

List of figures

Global Alignment ??
Local Alignment ??
Local Alignments to Detect Rearrangments ??
Bounded-Space Computation ??
Linear-Space Computation ??
Space-Saving Optimization ??
Divide and Conquer ??
Naive Karp-Rabin Algorithm ??
Efficiently Converting Strings to Numbers ??
Final Karp-Rabin Algorithm ??
The BLAST Algorithm ??
Nucleotide and amino acid match scores ??
3.1 Introduction

In the previous chapter, we used dynamic programming to compute sequence alignments in $O(n^2)$. In particular, you learned the Needleman-Wunsch algorithm for global alignment, which matches complete sequence with one another at the nucleotide level when they are known to be homologous (i.e. the sequences come from organisms that share a common ancestor).

The biological significance of finding sequence alignments is to be able to infer the most likely set of evolutionary events such as point mutations/mismatches and gaps (insertions or deletions) that occurred in order to transform one sequence into the other. By assigning costs to each transformation type (mismatch or gap) that reflect their respective levels of evolutionary difficulty, finding an optimal alignment reduces to finding the set of transformations that result in the lowest overall cost.

A dynamic algorithm, which uses optimal substructure to decompose a problem into similar sub-problems. The problem of finding a sequence alignment can be nicely expressed as a dynamic programming algorithm since alignment scores are additive, which means that finding the alignment of a larger sequence can be found by recursively finding the alignment to a smaller sequence. The scores are stored in a matrix, with one sequence corresponding to the columns and the other sequence corresponding to the rows. Each cell represents the transformation required between two nucleotides corresponding to the cell’s row and column. In contrast to a dynamic programming algorithm, a greedy algorithm simply chooses the transition with minimum cost at each step, which does not guarantee that the overall result will give the optimal or lowest-cost alignment. An alignment is recovered by tracing back through the dynamic programming matrix (shown below) in the global alignment algorithm.

![Figure 3.1: Global Alignment](image)

To summarize the Needleman-Wunsch algorithm for global alignment:

While computing scores corresponding to each cell in the matrix, we remember our choice (memoization) at that step i.e. which one of the top, left or diagonal cells led to the maximum score for the current cell. So, at the end, we are left with a matrix full of optimal scores at each cell position, along with pointers at each cell reflecting the optimal choice that leads to that particular cell.

The last step involves a traceback from the cell in the bottom right corner (which contains the score of aligning one complete sequence with the other) by following the pointers reflecting locally optimal choices, and then constructing the alignment corresponding to an optimal path followed in the matrix.

The running time of Needleman-Wunsch algorithm $O(n^2)$ since for each cell in the matrix, we do a finite amount of computation i.e. we calculate 3 values using already computed scores and then take the maximum of those values to find the score corresponding to that cell.

To guarantee correctness, it is necessary to compute the cost for every cell of the matrix. Strictly speaking, it is possible that the optimal alignment may be made up of a bad alignment (consisting of gaps and mismatches) at the start, followed by a really good alignment that makes it the best alignment overall. Since we do not know where the best alignment will come from, in theory, we need to compute every entry of the matrix to find a provable optimal global alignment. However, in practice, we can often restrict the alignment space to be explored, if we know that some alignments are clearly sub-optimal (this will be discussed in the current lecture). This is a heuristic, and while it may not work all the time since it is not a guaranteed optimal solution, it works remarkably well in practice. It was noted that depending on the properties of the scoring matrix, it may be possible to argue the correctness of
the bounded-space algorithm, even though such an algorithm will not be guaranteed to be faster all the time.

In this chapter, we will discuss database search for aligning a newly-sequenced gene to reference genes in a known genome. This chapter will also introduce the Smith-Waterman algorithm for local alignment for aligning subsequences as opposed to complete sequences, in contrast to the Needleman-Wunsch algorithm for global alignment. Later on in the chapter you will be given an overview of Hashing and semi-numerical methods like the Karp-Rabin algorithm for finding the longest (contiguous) common substring of nucleotides. These algorithms are implemented and extended for inexact matching in the BLAST program, one of the most famous and successful tools in computational biology. Finally, this chapter will go over BLAST as well as the probabilistic foundation of sequence alignment and how alignment scores can be interpreted as likelihood ratios.

Outline:

1. Global alignment vs. Local alignment
   - Variations on initialization, termination, update rule for Global Alignment (Needleman-Wunsch) vs. Local Alignment (Smith-Waterman)
   - Varying gap penalties, algorithmic speedups

2. Linear-time exact string matching
   - Karp-Rabin algorithm and semi-numerical methods
   - Hash functions and randomized algorithms

3. The BLAST algorithm and inexact matching
   - Hashing with neighborhood search
   - Two-hit blast and hashing with combs

4. Pre-processing for linear-time string matching
   - Fundamental pre-processing
   - Suffix Trees
   - Suffix Arrays
   - The Burrows-Wheeler Transform

5. Probabilistic foundations of sequence alignment
   - Mismatch penalties, BLOSUM and PAM matrices
   - Statistical significance of an alignment score

3.2 Global alignment vs. Local alignment

A global alignment is defined as the end-to-end alignment of two sequences.

A local alignment of string $s$ and $t$ is an alignment of a substring of $s$ with a substring of $t$.

In general, local alignments are used to find regions of high local similarity. Often, we are more interested in finding local alignments because we normally do not know the boundaries of genes and only a small domain of the gene may be conserved. In such a cases, we do not want to enforce that other (potentially non-homologous) parts of the sequence also align. Local alignment is also useful when searching for a small gene in a large chromosome. A local alignment is also useful in such a case where a long sequence may have undergone rearrangements and it may have been broken up into different smaller segments, since a local alignment could detect such rearrangements.

Another type of alignment is semi-global alignment.

This form of alignment is useful for overlap detection when we do not wish to penalize starting or ending gaps.
3.2.1 Using Dynamic Programming for local alignments

In this section we will see how to find local alignments with a minor modification of Needleman-Wunsch algorithm that was discussed in the previous chapter for finding global alignments.

To find global alignments, we use the following dynamic programming algorithm (Needleman-Wunsch algorithm):
Initialization : \( F(0,0) = 0 \)

\[ F(i,j) = \max \begin{cases} F(i-1,j) - d \\ F(i,j-1) - d \\ F(i-1,j-1) + s(x_i, y_j) \end{cases} \]

Termination : Bottom right

For finding local alignments we only need to modify the Needleman-Wunsch algorithm slightly to prevent an alignment score from going negative by checking and starting over to find a new local alignment. Since a local alignment can start anywhere, we initialize the first row and column in the matrix to zeros. Furthermore, since the alignment can end anywhere, we need to traverse the entire matrix to find the optimal alignment score (not only in the bottom right corner). The rest of the algorithm, including traceback, remains the unchanged.

To find local alignments we use the following dynamic programming algorithm for local alignment (Smith-Waterman algorithm):

\[ F(i,0) = 0 \]
\[ F(0,j) = 0 \]

\[ F(i,j) = \max \begin{cases} 0 \\ F(i-1,j) - d \\ F(i,j-1) - d \\ F(i-1,j-1) + s(x_i, y_j) \end{cases} \]

Termination : Anywhere

For finding a semi-global alignment, the important distinctions are to initialize the top row and leftmost column to zero and terminate end at either the bottom row or rightmost column.

To perform a semi-global alignment, the algorithm is as follows:

\[ F(i,0) = 0 \]
\[ F(0,j) = 0 \]

\[ F(i,j) = \max \begin{cases} F(i-1,j) - d \\ F(i,j-1) - d \\ F(i-1,j-1) + s(x_i, y_j) \end{cases} \]

Termination : Bottom Row or Right Column

### 3.2.2 Algorithmic Variations

Sometimes it can be costly in both time and space to run these alignment algorithms. Therefore, this section presents some algorithmic variations to save time and space that work well in practice.

One method to save time, which was brought up earlier, is the idea of limiting the space of alignments to be explored. Good alignments generally stay close to the diagonal of the matrix. Thus we can just explore a width of \( k \) around the diagonal. As mentioned at the beginning of this chapter, the problem with this modification is that this is a heuristic and can lead to a sub-optimal solution. Nevertheless, this works very well in practice. This algorithm requires \( O(k \times m) \) space and \( O(k \times m) \) time.

We saw earlier that in order to compute the optimal solution, we needed to store the score in each cell, as well as the pointer reflecting the optimal choice leading to each cell. However, if we are only interested in the optimal alignment score, and not the actual alignment itself, there is a method to compute the solution while saving space. To compute the score of any cell we only need the scores of the cell above, to the left,
and to the left-diagonal of the current cell. By saving the previous and current column in which we are computing scores, the optimal solution can be computed in linear space.

To compute the optimal alignment, however, we need quadratic time and linear space. The idea, based on the principle of divide-and-conquer, is that we compute the optimal alignments from both sides of the matrix i.e. from the left to the right, and vice versa. Let \( u = \lfloor \frac{n}{2} \rfloor \). Say we can identify \( v \) such that cell \((u, v)\) is on the optimal alignment path. That means \( v \) is the row where the alignment crosses column \( u \) of the matrix. We can find the optimal alignment by concatenating the optimal alignments from \((0, 0)\) to \((u, v)\) plus that of \((u, v)\) to \((m, n)\), where \( m \) and \( n \) is the bottom right cell (note: alignment scores of concatenated subalignments using our scoring scheme are additive. So we have isolated our problem to two separate problems in the the top left and bottom right corners of the DP matrix. Then we can recursively keep dividing up these subproblems to smaller subproblems, until we are down to aligning 0-length sequences or our problem is small enough to apply the regular DP algorithm. To find \( v \) the row in the middle column where the optimal alignment crosses we simply add the incoming and outgoing scores for that column.

One drawback of this divide-and-conquer approach is that it has a longer run-time. Nevertheless, the run-time is not dramatically increased. Since \( v \) can be found using one pass of regular DP, we can find \( v \) for each column in \( O(mn) \) time and linear space since we don’t need to keep track of traceback pointers for this step. Then by applying the divide and conquer approach, the subproblems take half the time since we only need to keep track of the cells diagonally along the optimal alignment path (half of the matrix of
the previous step) That gives a total run time of \( O(mn(1 + \frac{1}{2} + \frac{1}{4} + \ldots)) = O(2MN) = O(mn) \) (using the sum of geometric series), to give us a quadratic run time (twice as slow as before, but still same asymptotic behavior). The total time will never exceed \( 2MN \) (twice the time as the previous algorithm). Although the run-time is increased by a constant factor, one of the big advantages of the divide-and-conquer approach is that the space is dramatically reduced to \( O(N) \).

3.2.3 Generalized gap penalties

Gap penalties determine the score calculated for a subsequence and thus determine which alignment is selected. Depending on the model, it could be a good idea to penalize differently for, say, gaps of different lengths. However, the tradeoff is that there is also cost associated with using more complex gap penalty functions by substantially increasing running time. This cost can be mitigated by using simpler approximations to the gap penalty functions. The affine function is a fine intermediate: you have a fixed penalty to start a gap then a linear cost to add to a gap. You can also consider more complex functions that take into consideration the properties of protein coding sequences. In the case of protein coding regions alignment, a gap of length mod 3 can be less penalized.

3.3 Linear-time exact string matching

While we have looked at various forms of alignment and algorithms used to find such alignments, these algorithms are not fast enough for practical purposes. E.g. we may have a 100 nucleotide sequence which we want to search for in the whole genome, which may be over a billion nucleotides long. In this case, we want an algorithm with a run-time that depends on the length of query sequence, possibly with some pre-processing on the database, because processing the entire genome for every query would be extremely slow. For such problems, we enter the realm of randomized algorithms where instead of worrying about the worst-case performance, we are more interested in making sure that the algorithm is linear in the expected case. When looking for exact (consecutive) matches of a sequence, the Karp-Rabin algorithm interprets such a match numerically. There are many other solutions to this problem and some of them that can ensure the problem is linear in the worst case such as: the Z-algorithm, Boyer-Moore and Knuth-Morris-Pratt algorithm, algorithms based on suffix trees, suffix arrays, etc. (discussed in the “Lecture 3 addendum” slides)

3.3.1 Karp-Rabin Algorithm

This algorithm tries to match a particular pattern to a string this is the basic principle of database search. The problem is as follows: in text \( T \) of length \( n \) we are looking for pattern \( P \) of length \( m \). The key idea of the algorithm is that strings are mapped to numbers to enable fast comparison. A naive version of the algorithm involves mapping the string \( P \) and substrings of \( T \) of length \( m \) into numbers \( x \) and \( y \), respectively, sliding \( x \) along \( T \) at every offset until there is a match of the numbers.

However, one can see that the algorithm, as stated, is in fact non-linear for two reasons:

1. Computing each \( y_i \) takes more than constant time (it is in fact linear if we naively compute each number from scratch for each subsequence)
2. Comparing $x$ and $y_i$ can be expensive if the numbers are very large, which might happen if the pattern to be matched is very long.

To make the algorithm faster, we first modify the procedure for calculating $y_i$ in constant time by using the previously computed number, $y_i - 1$. We can do this using some bit operations: a subtraction to remove the high-order bit, a multiplication to shift the characters left, and an addition to append the low-order digit.

Also, to keep the numbers small to ensure efficient comparison, we do all our computations modulo $p$ (a form of hashing), where $p$ reflects the word length available to us for storing numbers. At the same time, $p$ is chosen in such a way that the comparison between $x$ and $y_i$ is not expensive. However, mapping to the space of numbers modulo $p$ can result in spurious hits due to hashing collisions, and so we modify the algorithm to deal with such spurious hits by explicitly verifying reported hits of the hash values. Hence, the final version of the Karp-Rabin algorithm is:

Questions:

Q4: What if there are more than 10 characters in the alphabet?

A4: In such a case, we can just modify the above algorithm by including more digits i.e. by working in a base other than 10, e.g. say base 256. But in general, when hashing is used, strings are mapped into a space of numbers and hence the strings are interpreted numerically.

Q5: Are there provisions in the algorithm for inexact matches?

A5: True, the above algorithm only works when there are regions of exact similarity between the query sequence and the database. In fact, the BLAST algorithm, which we look at later, extends the above
ideas to include the notion of searching in a biologically meaningful neighborhood of the query sequence to account for some inexact matches. This is done by searching in the database for not just the query sequence, but also some variants of the sequence up to some fixed number of changes.

In general, in order to reduce the time for operations on arguments like numbers or strings that are really long, it is necessary to reduce the number range to something more manageable. Hashing is a general solution to this and it involves mapping keys $k$ from a large universe $U$ of strings/numbers into a hash of the key $h(k)$ which lies in a smaller range, say $[1...m]$. There are many hash function that can be used, all with different theoretical and practical properties. The two key properties that we need are:

1. Reproducibility if $x = y$, then $h(x) = h(y)$. This is essential for our mapping to make sense
2. Uniform output distribution This implies that regardless of the input distribution, the output distribution is uniform. i.e. if $x \neq y$, then $P(h(x) = h(y)) = 1/m$, irrespective of the input distribution. This is a desirable property to reduce the chance of spurious hits.

An interesting idea that was raised was that it might be useful to have locality sensitive hash functions from the point of view of use in neighborhood searches, such that points in $U$ that are close to each other are mapped to nearby points by the hash function. The notion of Random projections, as an extension of the BLAST algorithm, is based on this idea. Also, it is to be noted that modulo doesn’t satisfy property 2 above because it is possible to have input distributions (e.g. all multiples of the number vis-à-vis which the modulo is taken) that result in a lot of collisions. Nevertheless, choosing a random number as the divisor of the modulo can avoid many collisions. Still, working with hashing increases the complexity of analyzing the algorithm since now we need to compute the expected run time by including the cost of verification. To show that the expected run time is linear, we need to show that the probability of spurious hits is small.

### 3.4 The BLAST algorithm (Basic Local Alignment Search Tool)

The BLAST algorithm looks at the problem of sequence database search, wherein we have a query, which is a new sequence, and a target, which is a set of many old sequences, and we are interested in knowing which (if any) of the target sequences is the query related to. One of the key ideas of BLAST is that it does not require the individual alignments to be perfect; once an initial match is identified, we can fine-tune the matches later to find a good alignment. Also, BLAST exploits a distinct characteristic of database search problems that most target sequences will be completely unrelated to the query sequence, and very few sequences will match. What this means is that correct (near perfect) alignments will have long substrings of nucleotides that match perfectly. E.g., if we looking for sequences of length 100 and are going to reject matches that are less than 90% identical, we need not look at sequences that do not even contain a consecutive stretch of 10 matching nucleotides in a row. In biology, the mutations that we find will not actually be distributed
randomly, but will be clustered in nonfunctional regions of DNA while leaving long stretches of functional DNA untouched. Therefore since highly similar sequences will have stretches of similarity, we can pre-screen the sequences for common long stretches; this idea is used in BLAST by breaking up the query sequence into \( W \)-mers and pre-screening the target sequences for all possible \( W \)-mers.

The other aspect of BLAST that allows us to speed up repeated queries is the ability to preprocess a large database of DNA off-line. After preprocessing, searching for a sequence of length \( m \) in a database of length \( n \) will take only \( O(m) \) time. The key insights that BLAST is based on are the ideas of hashing and neighborhood search that allows one to search for \( W \)-mers, even when there are no exact-matches.

### 3.4.1 The BLAST algorithm

The steps are as follows:

1. Split query into overlapping words of length \( W \) (the \( W \)-mers)
2. Find a “neighborhood” of similar words for each word (see below)
3. Lookup each word in the neighborhood in a hash table to find the location in the database where each word occurs. Call these the seeds, and let \( S \) be the collection of seeds.
4. Extend the seeds in \( S \) until the score of the alignment drops off below some threshold \( X \).
5. Report matches with overall highest scores

![Figure 3.13: The BLAST Algorithm](image)

The pre-processing step of BLAST is to make sure that all substrings of \( W \) nucleotides will be included in our database (or in a hash table). These are called the \( W \)-mers of the database. As in step 1, we first split the query by looking at all substrings of \( W \) consecutive nucleotides in the query. To find the neighborhood of these \( W \)-mers, we then modify these sequences by changing them slightly and computing their similarity to the original sequence. We generate progressively more dissimilar words in our neighborhood until our similarity measure drops below some threshold \( T \). This affords us to flexibility to find matches that do not have exactly \( W \) consecutive matching characters in a row, but which do have enough matches to be considered similar.

Then, we look up all of these words in our hash table to find seeds of \( W \) consecutive matching nucleotides. We then extend these seeds to find our alignment using the Smith-Waterman algorithm for local alignment, until the score drops below a certain threshold. Since the region we are considering is a much shorter segment, this will not be as slow as running the algorithm on the entire DNA database.
It is also interesting to note the influence of various parameters of BLAST on the performance of the algorithm vis-à-vis run-time and sensitivity:

- **W** Although large $W$ would result in fewer spurious hits/collisions, thus making it faster, there are also tradeoffs associated, namely: a large neighborhood of slightly different query sequences, a large hash table, and too few hits. On the other hand, if $W$ is too small, we may get too many hits which might make the algorithm slower.

- **T** If $T$ is higher, the algorithm will be faster, but you may miss sequences that are more evolutionarily distant. If comparing two related species, you can probably set a higher $T$ since you expect to find more matches between sequences that are quite similar.

- **X** Its influence is quite similar to $T$ in that both will control the sensitivity of the algorithm. While $W$ and $T$ affect the total number of hits one gets, and hence affect the run-time of the algorithm dramatically, setting a really stringent $X$ despite the more sensitive $W$ and $T$, will result in the algorithm being less sensitive and slower. So, it is important to match the stringency of $X$ with that of $W$ and $T$.

### 3.4.2 Extensions to BLAST

- **Filtering** Low complexity regions can cause spurious hits. For instance, if our query has a string of copies of the same nucleotide e.g. repeats of AC or just G, and the database has a long stretch of the same nucleotide, then there will be many many useless hits. To prevent this, we can either try to filter out low complexity portions of the query or we can ignore unreasonably over-represented portions of the database.

- **Two-hit BLAST** The idea here is to use double hashing wherein instead of hashing one long $W$-mer, we will hash two small $W$-mers. This allows us to find small regions of similarity since it is much more likely to have two smaller $W$-mers that match rather than one long $W$-mer. This allows us to get a higher sensitivity with a smaller $W$, while still pruning out spurious hits. This means that well spend less time trying to extend matches that dont actually match. Thus, this allows us to improve speed while maintaining sensitivity.

**Q6:** For a long enough $W$, would it make sense to consider more than 2 smaller $W$-mers?

**A6:** Before, it would be interesting to see how the number of such $W$-mers influences the sensitivity of the algorithm.

- **Combs** This is the idea of using non-consecutive $W$-mers for hashing. Recall from your biology classes that the third nucleotide in a triplet usually doesn’t actually have an effect on which amino acid is represented. This means that each third nucleotide in a sequence is less likely to be preserved by evolution, since it often doesn’t matter. Thus, we might want to look for $W$-mers that look similar except in every third codon. This is a particular example of a comb. A comb is simply a bit mask which represents which nucleotides we care about when trying to find matches. We explained above why 110110110 . . . might be a good comb, and it turns out to be. However, other combs are also useful. One way to choose a comb is to just pick some nucleotides at random. Rather than picking just one comb for a projection, it is possible to randomly pick a set of such combs and project the $W$-mers along each of these combs to get a set of lookup databases. Then, the query string can also be projected randomly along these combs to lookup in these databases, thereby increasing the probability of finding a match. This is called Random Projection. Extending this, an interesting idea for a final project is to think of different techniques of projection or hashing that make sense biologically.

### 3.5 Pre-processing for linear-time string matching

The hashing technique at the core of the BLAST algorithm is a powerful way of string for rapid lookup. A substantial time is invested to process the whole genome, or a large set of genomes, in advance of obtaining
a query sequence. Once the query sequence is obtained, it can be similarly processed and its parts searched
against the indexed database in linear time.

In this section, we briefly describe four additional ways of pre-processing a database for rapid string
lookup, each of which has both practical and theoretical importance.

3.5.1 Suffix Trees
Suffix trees provide a powerful tree representation of substrings of a target sequence T, by capturing all
suffixes of T in a radix tree.

Representation of a sequence in a suffix tree

**TODO: EXPAND** @scribe: Add image of suffix tree representation

Searching a new sequence against a suffix tree

**TODO: EXPAND** @scribe: Describe search procedure

Linear-time construction of suffix trees

**TODO: EXPAND** @scribe: The three key ideas for linear-time construction

3.5.2 Suffix Arrays
For many genomic applications, suffix trees are too expensive to store in memory, and more efficient repre-
sentations were needed to do so. Suffix arrays were developed specifically to reduce the memory consumption
of suffix trees, and achieve the same goals with a significantly reduced space need.

**TODO: EXPAND** @scribe: Add information on suffix arrays

3.5.3 The Burrows-Wheeler Transform
An even more efficient representation than suffix trees is given by the Burrows-Wheeler Transform, which
enables storing the entire hashed string in the same number of characters as the original string (and even
more compactly, as it contains frequent homopolymer runs of characters that can be more easily compressed).
This has helped make programs that can run even more efficiently.

**TODO: EXPAND** @scribe: Add example of compress-uncompress

This has had a very strong impact on short-string matching algorithms, and nearly all the fastest read
mappers are currently based on the Burrows-Wheeler Transform.

3.5.4 Fundamental pre-processing
A variation of processing that has theoretical interest but has found relatively little practical use in bioinfor-
matics. It relies on the Z vector, that contains at each position $i$ the length of the longest prefix of a string
that also matches the substring starting at $i$. This enables computing the $L$ and $R$ (Left and Right) vectors
that denote the end of the longest duplicate substrings that contains the current position $i$.

**TODO: EXPAND** @scribe: Use lecture from http://courses.csail.mit.edu/6.006/spring11/lectures/lec18-extra.pdf to expand this section

The Z algorithm enables an easy computation of both the Boyer-Moore and the Knuth-Morris-Pratt
algorithms for linear-time string matching.

3.6 Probabilistic Foundations of Sequence Alignment
As described above, the BLAST algorithm uses a scoring (substitution) matrix to expand the list of $W$-mers
to look for and to determine an approximately matching sequence during seed extension. Also, a scoring
matrix is used in evaluating matches or mismatches in the alignment algorithms. But how do we construct this matrix in the first place? How do you determine the value of \( s(x_i, y_j) \) in global/local alignment?

The idea behind the scoring matrix is that the score of alignment should reflect the probability that two similar sequences are homologous i.e. the probability that two sequences that have a bunch of nucleotides in common also share a common ancestry. For this, we look at the likelihood ratios between two hypotheses

1. **Hypothesis 1:** – That the alignment between the two sequence is due to chance and the sequences are, in fact, unrelated.

2. **Hypothesis 2:** – That the alignment is due to common ancestry and the sequences are actually related.

Then, we calculate the probability of observing an alignment according to each hypothesis. \( Pr(x, y|U) \) is the probability of aligning \( x \) with \( y \) assuming they are unrelated, while \( Pr(x, y|R) \) is the probability of the alignment, assuming they are related. Then, we define the alignment score as the log of the likelihood ratio between the two:

\[
S \equiv \log \frac{Pr(x, y|R)}{Pr(x, y|U)}
\]

So, when we add up the scores of individual alignments at each position in the alignment of two sequences, since a sum of logs is a log of a product, we get that the total score of the alignment gives the probability of the whole alignment, assuming each individual alignment is independent. Thus, an additive matrix score exactly gives us the probability that the two sequences are related, and the alignment is not due to chance. More formally, considering the case of aligning proteins, for unrelated sequences, the probability of having an \( n \)-residue alignment between \( x \) and \( y \) is a simple product of the probabilities of the individual sequences since the residue pairings are independent. That is,

\[
\begin{align*}
  x &= \{x_1 \ldots x_n\} \\
  y &= \{y_1 \ldots x_n\} \\
  q_a &= P(\text{amino acid } a) \\
  Pr(x, y|U) &= \prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{n} q_{y_i}
\end{align*}
\]

For related sequences, the residue pairings are no longer independent so we must use a different joint probability, assuming that each pair of aligned amino acids evolved from a common ancestor:

\[
\begin{align*}
  p_{ab} &= P(\text{evolution gave rise to } a \text{ in } x \text{ and } b \text{ in } y) \\
  Pr(x, y|R) &= \prod_{i=1}^{n} p_{x_i, y_i}
\end{align*}
\]
Then, the likelihood ratio between the two is given by:

\[
P(x, y|R) = \frac{\prod_{i=1}^{n} p_{x_i y_i}}{\prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{n} q_{y_i}}
\]

Since we eventually want to compute a sum of scores and probabilities require add products, we take the log of the product to get a handy summation:

\[
S \equiv \log \frac{P(x, y|R)}{P(x, y|U)} = \sum_i \log \left( \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}} \right) = \sum_i s(x_i, y_i)
\]

Thus, the substitution matrix score for a given pair \(a, b\) is give by

\[
s(a, b) = \log \left( \frac{p_{ab}}{q_{a} q_{b}} \right)
\]

The above expression is then used to crank out a substitution matrix like the BLOSUM62 for amino acids. It is interesting to note that the score of a match of an amino acid with itself depends on the amino acid itself because the frequency of random occurrence of an amino acid affects the terms used in calculating the likelihood ratio score of alignment. Hence, these matrices capture not only the sequence similarity of the alignments, but also the chemical similarity of various amino acids.

### 3.7 Current Research Directions

### 3.8 Further Reading

### 3.9 Tools and Techniques

### 3.10 What Have We Learned?

Bibliography
CHAPTER

FOUR

COMPARATIVE GENOMICS I: GENOME ANNOTATION

Mark Smith, Yarden Katz
(Partially adapted from notes by:
Angela Yen, Christopher Robde, Timo Somervuo and Saba Gul)

List of figures

Comparative identification of functional elements in 12 Drosophila species ??
Comparative genomics reveals functional elements and evolutionary signatures ??
RNA with secondary stem-loop structure ??
MicroRNA with sections of stem-loop complementary to mRNA ??
Some point mutations to DNA sequence do not change protein translation ??
Rate matrix $Q_c$ estimated from known coding regions ??
Rate matrix $Q_N$ estimated from non-coding regions ??
Evolutionary signatures can predict new genes and exons ??
Codon substitution rate matrices ??
Signatures of protein coding vs. non-coding regions ??
OPRL1 neurotransmitter ??
Various interpretations of stop codon suppression ??
Z-curve for Caki. ??
Genomic regions show more conservation than expected ??
Null model rate matrix ??
Rate of synonymous and non-synonymous substitutions ??
Hairpin structure of a microRNA ??
Characteristic conservation pattern of miRNAs ??
Certain sequence motifs are conserved in different functional elements ??
4.1 Introduction

In this lecture we will explore the emerging field of comparative genomics, primarily through examples of multiple species genome alignments (work done by the Kellis lab.) One approach to the analysis of genomes is to infer important gene functions using an understanding of evolution to search for expected evolutionary patterns. This will be the focus of the current lecture. Another approach is to discover evolutionary trends by studying genomes themselves. This is the focus of the next lecture. Taken together, evolutionary insight and large genomic datasets offer great potential for discovery of novel biological phenomena.

A recurring theme of this work is that by taking a global computational approach to analyzing elements of genes and RNAs encoded in the genome, one can find interesting new biological phenomena by seeing how individual examples “diverge” or differ from the average case. For example, by examining many protein-coding genes, we can identify features representative of that class of loci, and then come up with highly accurate tests for distinguishing protein-coding from non-protein-coding genes. Often, these computational tests, based on thousands of examples, will be far more definitive than conventional low-throughput wet lab tests (such as mass spectrometry to detect protein products, in cases where we want to know if a particular locus is protein coding.)

4.1.1 Motivation

As the cost of genome sequencing continues to drop, the availability of sequenced genome data has exploded. Comparative genomics leverages this power to look for biologically relevant signatures. Many important biological phenomena are difficult to discover through experimental work alone, but can be flagged for further study through genome analysis. For example, CRISPRs, which provide microbial adaptive immunity, were only recently discovered through motif analysis and later experimental confirmation. Without knowing where to look, experimentalists overlooked this important feature for decades. Given the prevalence of sequences lacking a well-understood function, it is likely that many more important biological phenomena are waiting to be discovered.

This is a particularly exciting time to enter the field of comparative genomics, because the field is mature enough that there are tools and data available to make discoveries but young enough that important findings will likely continue to be made for many years. Comparative genomics complements experimental work by building on previously understood biological mechanisms to develop and test expected models of sequence evolution. By simultaneously quantitatively evaluating many diverse organisms, comparative genomics offers a more comprehensive and systematic view of evolution than might be possible through other methods of interrogating biological systems. Unsupervised learning techniques enable the discovery of important features even without strong a priori understanding of the phenomena being studied, an opportunity without a strong experimental parallel. Comparative genomics offers more than just a survey of evolution, however, it also enables the discovery of otherwise cryptic molecular mechanisms such as stop-codon read-through. Once identified in silico, these features can then be confirmed through in vivo experiments. Importantly, through the implementation of scalable algorithms, genomic research can advance at a much faster speed and lower cost than experimental work.

4.1.2 The Challenge

An endless string of ATGCs representing the sequence of all DNA on earth is meaningless without a system for interpreting and identifying important features in this sea of data. An approach naive to biology might miss the signatures of synonymous substitutions or frame shift mutations, while an approach naive to computer science might be hopelessly inefficient when faced with the ever larger datasets emerging from sequencing centers. Comparative genomics requires rare multidisciplinary skills and insight.

4.1.3 Importance of many closely-related genomes

In order to resolve significant biological features it is necessary to have both sufficient similarity to enable comparison and sufficient divergence to identify signatures of change over evolutionary time. It is difficult to achieve both similarity and divergence in a pairwise comparison. By extending analysis to many genomes
simultaneously with some clusters of similar organisms and some dissimilar organisms, we improve the
resolution of our analysis.

A simple analogy is one of observing an orchestra. If you place a single microphone, it will be difficult to
decipher the signal coming from the entire system, because it will be overwhelmed by the local noise from
the single point of observation, the nearest instrument. If you place many microphones distributed across
the orchestra at reasonable distances, then you get a much better perspective not only on the overall signal,
but also on the structure of the local noise.

Similarly, by sequencing many genomes across the tree of life we are able to distinguish the biological
signals of functional elements from the noise of neutral mutations. This is because nature selects for conser-
vation of functional elements across large phylogenetic distances while constantly introducing noise through
mutagenic processes operating at shorter distances.

4.1.4 Data availability assumption: Whole-Genome Alignments and Assemblies

In this chapter, we will assume that we already have a complete genome-wide alignment of multiple closely-
related species, spanning both coding and non-coding regions. In practice, constructing complete genome
assemblies and whole-genome alignments is a very challenging problem, that will be the topic of the next
chapter.

4.2 Conservation of functional elements

4.2.1 Conservation of functional elements in Drosophila

In a 2007 paper (slide 8)\(^1\), evolutionary signatures of different functional elements and predicts function using
conserved signatures. One important finding illustrated below (from slide 12) is that across evolutionary
time, genes tend to remain in a similar location. This is shown by the multiple alignment below, where
genes mostly stay in the same position (indicated by vertical lines) or remain in the same neighborhood
(diagonal lines. Moreover, by comparing these genes to a reference genome (ie human or model of interest)
it is possible to observe the level of conservation across different regions.

But how do we know what parts of our genome have been conserved by evolution? Given a genome,
we can align it with the genomes of other evolutionarily-related species. We BLAST regions of the genome
against other genomes, and then align individual regions to a reference genome for a pair of species. We
then perform a series of pairwise alignments walking up the phylogeny until we have an alignment for all
sequences. Because we can align every gene and every intergenic region, we don’t just have to rely on
conserved regions, we can align every single region regardless of whether the conservation in that region is
sufficient to allow genome wide placement across the species. This is because we have ‘anchors’ spanning
entire regions, and we can thus infer that the entire region in conserved as a block and then apply global
alignment to the block.

4.2.2 Diversity of evolutionary signatures

In the figure below, we see a DNA sequence represented on the x-axis, while each row represents a different
species. The y-axis within each row represents the amount of conservation for that species in that part of the
chromosome (though other species that are not shown were also used to calculate conservation). To calculate
the degree of conservation, a Hidden Markov Model (HMM) was used with two states: high conservation
and low conservation. The amount of conservation comes from the score calculated using posterior decoding
of the high conservation model. Essentially, the amount of conservation represents the probability of being
in the conserved state of the model at this locus. From this figure, we can see that there are blocks of
conservation separated by regions that are not conserved. The 12 exons (highlighted by red rectangles) are
mostly conserved across species, but sometimes, certain exons are missing; for example, zebrafish is missing
exon 9. However, we also see that there is a spike in some species (as circled in red) that do not correspond
to a known protein coding gene. This tells us that some intronic regions have also been evolutionarily

\(^1\)http://www.nature.com/nature/journal/v450/n7167/abs/nature06340.html
conserved, since DNA regions that do not code for proteins can still be important as functional elements, such as RNA, microRNA, and regulatory motifs. By observing how regions are conserved, instead of just looking at the amount of conservation, we can observe evolutionary signatures of conservation for different functional elements. That is because different functional elements have different selective pressures (due to their structure and other characteristics); some changes (insertions, deletions, or mutations) that can be extremely harmful to one functional element may be innocuous to another. By figuring out what the signatures are for different elements, we can more accurately annotate a region by seeing what patterns of conservation it shows.

Importantly, there is distinct phylogenetic structure to the pattern of conservation. More similar species (mammals) group together with shared conserved domains that fish lack, suggesting a mammalian specific innovation, perhaps for regulatory elements not shared by fish. Meanwhile, some features are globally conserved, suggesting a universal significance, such as protein coding. Initial approximate annotation of protein coding regions in the human genome was possible using the simple heuristic that if it was conserved from human to fish it likely served as a protein coding region.

An interesting idea for a final project would be to map divergences in the multiple alignment and call these events births of new coding elements. By focusing on a particular element (say microRNAs) one could identify periods of innovation and isolate portions of a phylogenetic tree enriched for certain classes of these elements.

Different evolutionary signatures exist for the different functional elements. The rest of the lecture will focus on quantifying the degree to which a sequence follows a given pattern. Kellis compared the process of evolution to exploring a fitness landscape, with the fitness score of a particular sequence constrained by the function it encodes. For example, protein coding genes are constrained by selection on the translated product, so synonymous substitutions in the third base pair of a codon are tolerated. Below is a summary of the expected patterns followed by various functional elements as illustrated on slide 17.

- Protein-coding genes exhibit particular frequencies of codon substitution as well as reading frame conservation. This makes sense because the significance of the genes is the proteins they code for;
therefore, changes that result in the same or similar amino acids can be easily tolerated, while a tiny change that drastically changes the resulting protein can be considered disastrous. In addition to the error correction of the mismatch repair system and DNA polymerase itself, the redundancy of the genetic code provides an additional level of intrinsic error correction/tolerance.

- Structural RNA is selected based on the secondary sequence of the transcribed RNA, and thus requires compensatory changes. For example, some RNA has a secondary stem-loop structure such that sections of its sequence bind to other sections of its sequence in its stem, as shown below.

Imagine that a nucleotide (A) and its partner (T) bind to each other in the stem, and then both mutate to a C and G, respectively. Since they are still complementary, this is a compensatory change that maintains its secondary structure. However, if just the T mutated to an A, they would no longer be complementary, so this mutation would not be maintained by evolution, as it ruins the secondary structure. Therefore, in RNA structures, the amount of change to the secondary structure (e.g. stem-loop) is more important than the amount of change in the primary structure (just the sequence). Understanding the effects of changes in RNA structure requires knowledge of the secondary structure. The likely secondary structure of an RNA can be determined by modeling the stability of many possible conformations and choosing the most likely conformation.

- MicroRNA is a molecule that is ejected from the nucleus into the cytoplasm. Their characteristic trait is that they are also have the hairpin (stem-loop) structure, but a section of the stem (highlighted in red and blue in the figure below) is complementary to a portion of mRNA.

When microRNA binds its complementary sequence to the respective portion of mRNA, it degrades the mRNA. This means that it is a post-transcriptional regulator, since its being used to limit the production of a protein (translation) after transcription. MicroRNA is conserved differently than structural RNA. Due to its binding to an mRNA target, the region of binding is much more conserved to maintain target specificity.

- Finally, regulatory motifs are conserved in sequence (to bind particular interacting protein partners) but not necessarily in location. Regulatory motifs can move around since they only need to recruit a factor to a particular region. Small changes (insertions and deletions) that preserve the consensus of the motif are tolerated, as are changes upstream and downstream that move the location of the motif.

When trying to understand the role of conservation in functional class prediction, an important question is how much of observed conservation can be explained by known patterns. Even after accounting for random conservation, roughly 60% of non-random conservation in the fly genome was not accounted for—that is, we couldn’t identify it as a protein-coding gene, RNA, microRNA, or regulatory motif. The fact that they remain
conserved however suggests a functional role. That so much conserved sequence remains poorly understood underscores that many exciting questions remain to be answered. One final project for 6.047 in the past was using clustering (unsupervised learning) to account for the other conservation. It developed into an M.Eng project, and some clusters were identified, but the function of these clusters was, and is, still unclear. Its an open problem!

### 4.3 Protein-Coding Signatures

In slide 12, we see three examples of conservation: an intronic sequence with poor conservation, a coding region with high conservation, and a non-coding region with high conservation, meaning it is probably a functional element. The important characteristic of protein-coding regions to remember is that codons (triples of nucleotides) code for amino acids, which make up proteins. This results in the evolutionary signature of protein-coding regions, as shown in slide 13: (i) reading-frame conservation and (ii) codon-substitution patterns.

The intuition for this signature is relatively straightforward. Firstly, reading frame conservation makes sense, since an insertion or deletion of one or two nucleotides will shift how all the following codons are read. However, if an insertion or deletion happens in a multiple of 3, the other codons will still be read in the same way, so this is a less significant change. Secondly, it makes sense that some mutations are less harmful than others, since different triplets can code for the same amino acids (a conservative substitution, as evident from the matrix below), and even mutations that result in a different amino acid may be evolutionarily neutral if the substitutions occur with similar amino acids in a domain of the protein where exact amino acid properties are not required. These distinctive patterns allow us to color the genome and clearly see where the exons are, as shown in slides 14-16.

![Figure 4.5: Some point mutations to DNA sequence do not change protein translation](image)

#### 4.3.1 Reading-Frame Conservation (RFC)

By scoring the pressure to stay in the same reading frame we can quantify how likely a region is to be protein-coding or not. As shown in slide 20, we can do this by having a target sequence (Scer), and then aligning a selecting sequence (Spar) to it and calculating what proportion of the time the selected sequence matches the target sequences reading frame. Since we dont know where the reading frame starts in the selected sequence, we align three times (Spar\_1, Spar\_2, Spar\_3) to try all possible offsets, and then choose
the alignment where the selected sequence is most often in sync with the target sequence. Finally, for the
best alignment, we calculate the percentage of nucleotides that are out of frame if it is above a cutoff,
this selected species votes that this region is a protein-coding region, and if it is low, this species votes
that this is an intergenic region. The votes are tallied from all the species to sum to the RFC score. This
method is not robust to sequencing error. We can compensate for these errors by using a smaller scanning
window and observing local reading frame conservation. This method was shown to have 99.9% specificity
and 99% sensitivity when applied to the yeast genome. When applied to 2000 hypothetical ORFs (open
reading frames, or proposed genes) in yeast, it rejected 500 of these putative protein coding genes as not
being protein coding. Similarly, 4000 hypothetical genes in the human genome were rejected by this method.
This model created a specific hypothesis (that these DNA sequences were unlikely to code for proteins) that
has subsequently been supported with experimental confirmation that the regions do not code for proteins in
vivo. This represents an important step forward for genome annotation, because previously it was difficult
to conclude that a DNA sequence was non-coding simply from lack of evidence. By narrowing the focus and
creating a null hypothesis (that the gene in question appears to be a non-coding gene) it became much
easier to not only accept coding genes, but to reject non-coding genes with computational support.

During the discussion of reading frame conservation in class, we identified an exciting idea for a final
project which would be to look for the birth of new functional proteins resulting from frame shift mutations.

4.3.2 Codon-Substitution Frequencies (CSFs)

The second signature of protein coding regions, the codon substitution frequencies, acts on multiple levels
of conservation. To explore these frequencies, it is helpful to remember that codon evolution can be modeled
by conditional probability distributions (CPDs) the likelihood of a descendant having a codon b where an
ancestor had codon a an amount of time t ago. The most conservative event is exact maintenance of the
codon. A mutation that codes for the same amino acid is conservative but not totally synonymous, because
of species specific codon usage biases. Even mutations that alter the identity of the amino acid might be
conservative if they code for amino acids with similar biochemical properties. We use a CPD in order to
capture the net effect of all of these considerations.

To calculate these CPDs, we need a rate matrix, Q, which measures the exchange rate for a unit time;
that is, it indicates how often codon a in species 1 is substituted for codon b in species 2, for a unit branch
length. Then, by using $e^{Qt}$, we can estimate the frequency of substitution at time t. The intuition, as shown
in slide 31, is that as time increases, the probability of substitutions increase, while at the initial time ($t = 0$),
$e^{Qt}$ is the identity matrix, since every codon is guaranteed to be itself.

But how do we get the rate matrix? Q is learned from the sequences, by using Expectation-Maximization,
for example. Given the parameters of the model, we can use Felsensteins algorithm to compute the probability
of any alignment, while taking into account phylogeny, given the substitution model (the E-step). Then,
given the alignments and phylogeny, we can choose the parameters (the rate matrix: Q, and branch lengths:
t) that maximize the likelihood of those alignments in the M-step; for example, to estimate Q, we can count
the number of times one codon is substituted for another in the alignment.

Now that we know how to obtain our model, we note that, given the specific pattern of codon substitution
frequencies for protein-coding, we want two models so that we can distinguish between coding and non-coding
regions. The images below shows the two distinct rate matrices, one each for genes and intergenic regions,
where a lighter color means the substitution is more likely. A number of salient features present themselves
in the codon substitution matrix (CSM) for genes. Note that the main diagonal element has been removed,
because the frequency of a triplet being exchanged for itself will obviously be much higher than any other
exchange. Nevertheless,

1. it is immediately obvious that there is a strong diagonal element in the protein coding regions.

2. We also note certain high-scoring off diagonal elements in the coding CSM: these are substitutions that
   are close in function rather than in sequence, such as 6-fold degenerate codons or very similar amino
   acids.

---


3. We also note dark vertical stripes, which indicate these substitutions are especially unlikely. These columns correspond to stop codons, since substitutions to this triplet would significantly alter protein function, and thus are strongly selected against.

On the other hand, in the matrix for intergenic regions, the exchange rates are more uniform. In these regions, what matters is the mutational pattern, i.e. whether a change is one or more mutations away. Therefore, intergenic regions are dictated by mutational proximity whereas genetic regions are dictated by selective proximity.

![Figure 4.6: Rate matrix \( Q_c \) estimated from known coding regions](image)

![Figure 4.7: Rate matrix \( Q_N \) estimated from non-coding regions](image)

Now that we have the two rate matrices for the two regions, we can calculate the probabilities that each matrix generated the genomes of the two species, as shown in slide 28. This can be done by using Felsenstein’s algorithm, and adding up the score for each pair of corresponding codons in the two species. Finally, we can calculate the likelihood ratio that the alignment came from a coding region to a non-coding region by dividing the two scores: this demonstrates our confidence in our annotation of the sequence. If the ratio is greater than 1, we can guess that it is a coding region, and if it is less than 1, then it is a non-coding region. For example, in slide 29, we are very confident about the respective classifications of each region.

It should be noted, however, that although the coloring of the sequences on this slide confirms our classifications, the likelihood ratios are calculated independently of the coloring, which uses our knowledge of synonymous or conservative substitutions. This further implies that this method automatically infers the genetic code from the pattern of substitutions that occur, simply by looking at the high scoring substitutions. In species with a different genetic code, the patterns of codon exchange will be different; for example, in Candida albumin, the CTG codes for serine (polar) rather than leucine (hydrophobic), and this can be deduced from the CSMs. However, no knowledge of this is required by the method; instead, we can deduce this \textit{a posteriori} from the CSM.

### 4.3.3 Classification of \textit{Drosophila} Genome Sequences

On slide 31, we see that using these RFC and CSF metrics allows us to classify exons and introns with extremely high specificity and sensitivity. The classifiers that use these measures to classify sequences can be implemented using a HMM or semi-Markov conditional random field (SMCRF). CRFs allows the
integration of diverse features that do not necessarily have a probabilistic nature, whereas HMMs require us to model everything as transition and emission probabilities. CRFs will be discussed in an upcoming lecture. One might wonder why these more complex methods need to be implemented, when the simpler method of checking for conservation of the reading frame worked well. The reason is that in very short regions, insertions and deletions will be very infrequent, even by chance, so there won’t be enough signal to make the distinction between protein-coding and non-protein-coding regions. In the figure below, we see a DNA sequence along the x-axis, with the rows representing an annotated gene, amount of conservation, amount of protein-coding evolutionary signature, and the result of Viterbi decoding using the SMCRF, respectively.

Figure 4.8: Evolutionary signatures can predict new genes and exons

This is one example of how utilization of the protein-coding signature to classify regions has proven very successful. Identification of regions that had been thought to be genes but that did not have high protein-coding signatures allowed us to strongly reject 414 genes in the fly genome previously classified as CGid-only genes, which led FlyBase curators to delete 222 of them and flag another 73 as uncertain. In addition, there were also definite false negatives, as functional evidence existed for the genes under examination. Finally, in the data, we also see regions with both conservation, as well as a large protein-coding signature, but had not been previously marked as being parts of genes, as in the figure above. Some of these have been experimentally tested and have been show to be parts of new genes or extensions of existing genes. This underscores the utility of computational biology to leverage and direct experimental work.

4.4 microRNA (miRNA) genes

One example where excess constraint is observed is in regions encoding micro RNAs (miRNA). miRNAs are RNA molecules that bind to complementary sequences in the 3’ untranslated region of targeted mRNA molecules, causing gene silencing. How do we find evolutionary signatures for miRNA genes and their targets, and can we use these to gain new insights on their biological functions? We will see that this is a challenging task, as miRNAs leave a highly conserved but very subtle evolutionary signal.
4.4.1 Computational Challenge
Predicting the location of miRNA genes and their targets is a computationally challenging problem. We can look for “hairpin” regions, where we find nucleotide sequences that are complementary to each other and predict a hairpin structure. But out of 760,355 miRNA-like hairpins found in the cell, only 60-100 were true miRNAs. So to make any test that will give us regions statistically likely to be miRNAs, we need a test with 99.99% specificity.

Below is an example of the conservation pattern for miRNA genes. You can see the two hairpin structures conserved in the red and blue regions, with a region of low conservation in the middle. This pattern is characteristic of miRNAs. By analyzing evolutionary and structural features specific to miRNA, we can use combinations of these features to pick out regions of miRNAs with > 4,500 enrichment compared to random hairpins. The following are examples of features that help pick out miRNAs:

- miRNAs bind to highly conserved target motifs in the 3’ UTR
- miRNAs can be found in introns of known genes
- miRNAs have a preference for the positive strand of DNA and for transcription factors
- miRNAs are not found in exonic and repetitive elements of the genome

Applying such a test to the fly genome showed 101 hairpins above the 0.95 cutoff, rediscovering 60 of 74 of known miRNAs, predicting 24 novel miRNAs that were experimentally validated, and finding an additional 17 candidates that showed evidence of diverse function.

4.5 Regulatory Motifs
Another class of functional element that is highly conserved across many genomes contains regulatory motifs. A regulatory motif is a highly conserved sequence of nucleotides that occurs many times throughout the genome and serves some regulatory function. For instance, these motifs might characterize enhancers, promoters, or other genomic elements.

4.5.1 Computationally Detecting Regulatory Motifs
Computational methods have been developed to measure conservation of regulatory motifs across the genome, and to find new unannotated motifs de novo.

Known motifs are often found in regions with high conservation, so we can increase our testing power by testing for conservation, and then finding signatures for regulatory motifs.
Figure 4.11: Certain sequence motifs are conserved in different functional elements of genomes.

Evaluating the pattern of conservation for known motifs versus the “null model” of regions without motifs gives the following signature:

<table>
<thead>
<tr>
<th>Conservation within:</th>
<th>Gal4 (known motif region)</th>
<th>Controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>All intergenic regions</td>
<td>13%</td>
<td>2%</td>
</tr>
<tr>
<td>Intergenic: coding</td>
<td>13%: 3%</td>
<td>2%:7%</td>
</tr>
<tr>
<td>Upstream: downstream</td>
<td>12:0</td>
<td>1:1</td>
</tr>
</tbody>
</table>

So as we can see, regions with regulatory motifs show a much higher degree of conservation in intergenic regions and upstream of the gene of interest.

To discover novel motifs, we can use the following pipeline:

- Pick a motif “seed” consisting of two groups of three non-degenerate characters with a variable size gap in the middle.
- Use a conservation ratio to rank the seed motifs.
- Expand the seed motifs to fill in the bases around the seeds using a hill climbing algorithm.
- Cluster to remove redundancy.

This algorithm is given in more detail on page 12 in the first set of lecture 5 slides.

Discovering motifs and performing clustering has led to the discovery of many motif classes, such as tissue specific motifs, function specific motifs, and modules of cooperating motifs.

### 4.6 Individual Instances of Regulatory Motifs

To look for expected motif regions, we can first calculate a branch-length score for a region suspected to be a regulatory motif, and then use this score to give us a confidence level of how likely something is to be a real motif.

The branch length score (BLS) sums evidence for a given motif over branches of a phylogenetic tree. Given the pattern of presence or absence of a motif in each species in the tree, this score evaluates the total branch length of the sub-tree connecting the species that contain the motif. If all species have the motif, the BLS is 100%. Note more distantly related species are given higher scores, since they span a longer evolutionary distance. If a predicted motif has spanned such a long evolutionary time frame, it is likely it is a functional element rather than just a region conserved by random chance.

To create a null model, we can choose control motifs. The null model motifs should be chosen to have the same composition of the original motif, to not be too similar to each other, and to be dissimilar from known motifs. We can get a confidence score by comparing the fraction of motif instances to control motifs at a given BLS score.
More information on specific motifs and methods of detecting motifs can be found on pages 14 and 15 of the slides.

4.7 Individual Nucleotides Under Selection

As mentioned previously, analysis and detection of individual nucleotide selection can be useful for several applications. Due to amino acid codon degeneracy, single nucleotide polymorphisms (SNPs) within coding regions will not necessarily change the amino acid sequence produced. However, when it does affect the codon, they can result in a different amino acid being coded or a premature stop being induced.

4.7.1 Detecting Rates and Patterns of Selection

Rates of nucleotide selection can be detected by estimating the intensity of constraint from a probabilistic model of substitution rate. The intensity of constraint can be obtained by using a Maximum Likelihood (ML) estimation to fit a statistical model to the sequence data and estimating the model parameters. Unusual substitution patterns are detected on basis of finding a probabilistic model with a stationary distribution different from that of the background.

Based on the analysis, an estimate of the portion of the genome under constraint can be obtained and $X\%$ under constraint means that a randomly chosen k-mer in the genome has a $X/100$ probability of being under selection.

4.8 Biological Findings and Surprises

4.8.1 Unusual Protein-Coding Genes

We saw in the previous section how we could distinguish between non-coding and coding regions of the genome based on their evolutionary signatures. We did this by creating two separate 64 by 64 rate matrices: one measuring the rate of codon substitutions in coding regions, and the other in non-coding regions. The rate matrix gives the exchange rate of codons or nucleotides over a unit time.

We used the two matrices to calculate two probabilities for any given alignment: the likelihood that it came from a coding region and the likelihood that it came from a non-coding region. Taking the likelihood ratio of these two probabilities gives a measure of confidence that the alignment is protein-coding. Using this method we can pick out regions of the genome that evolve according to the protein coding signature. We will see later how to combine this likelihood ratio approach with phylogenic methods to find evolutionary

Figure 4.12: Signatures of protein coding vs. non-coding regions. Protein coding regions (left) show a high degree of synonymous substitutions (in green, substitutions which do not change the amino acid a codon encodes). Non-protein coding regions, on the other hand, show a high degree of non-conservative regions.

However, this method only lets us find regions that are selected at the translational level. The key point is that here we are measuring for only protein coding selection. We will see today how we can look for other conserved functional elements that exhibit their own unique signatures.
4.8.2 Leaky Stop Codons

Stop codons (TAA, TAG, TGA in DNA and UAG, UAA, UGA in RNA) typically signal the end of a gene. They clearly reflect translation termination when found in mRNA and release the amino acid chain from the ribosome. However, in some unusual cases, translation is observed beyond the first stop codon. In instances of single read-through, there is a stop codon found within a region with a clear protein-coding signature followed by a second stop-codon a short distance away. An example of this in the human genome is given in Figure ?? . This suggests that translation continues through the first stop codon. Instances of double read-through, where two stop codons lie within a protein coding region, have also been observed. In these instances of stop codon suppression, the stop codon is found to be highly conserved, suggesting that these skipped stop codons play an important biological role.

Translational read-through is conserved in both flies, which have 350 identified proteins exhibiting stop codon read-through, and humans, which have 4 identified instances of such proteins. They are observed mostly in neuronal proteins in adult brains and brain expressed proteins in Drosophila.

Figure 4.13: OPRL1 neurotransmitter: one of the four novel translational read-through candidates in the human genome. Note that the region after the first stop codon exhibits an evolutionary signature similar to that of the coding region before the stop codon, indicating that the stop codon is “suppressed”.
The kelch gene exhibits another example of stop codon suppression at work. The gene encodes two ORFs with a single UGA stop codon between them. Two proteins are translated from this sequence, one from the first ORF and one from the entire sequence. The ratio of the two proteins is regulated in a tissue-specific manner. In the case of the kelch gene, a mutation of the stop codon from UGA to UAA results in a loss of function, suggesting that tRNA suppression is the mechanism behind stop codon suppression.

An additional example of stop codon suppression is, Caki, a protein active in the regulation of neurotransmitter release in Drosophila. Open reading frames (ORFs) are DNA sequences which contain a start and stop codon. In Caki, reading the gene in the first reading frame (Frame 0) results in significantly more ORFs than reading in Frame 1 or Frame 2 (a 440 ORF excess). Figure ?? lists twelve possible interpretations for the ORF excess. However, because the excess is observed only in Frame 0, only the first 4 interpretations are likely:

- **Stop-codon readthrough:** the stop codon is suppressed when the ribosome pulls in tRNA that pairs incorrectly with the stop codon.

- **Recent nonsense:** Perhaps some recent nonsense mutation is causing stop codon readthrough.

- **A to I editing:** Unlike we previously thought, RNA can still be edited after transcription. In some case the A base is changed to an I, which can be read as a G. This could change a TGA stop codon to a TGG, which encodes an amino acid. However, this phenomenon is only found in a couple of cases.

- **Selenocysteine, the “21st amino acid”:** Sometimes when the TGA codon is read by a certain loop which leads to a specific fold of the RNA, it can be decoded as selenocysteine. However, this only happens in four fly proteins, so can’t explain all of stop codon suppression.

Among these four, three of them (recent nonsense, A to I editing, and selenocysteine) account for only 17 of the cases. Hence, it seems that read-through must be responsible for all if not most of the remaining cases. In addition, biased stop codon usage is observed hence ruling out other processes such as alternative splicing (where RNA exons following transcription are reconnected in multiple ways leading to multiple proteins) or independent ORFs.
Read-through regions can be determined in a single species based on their pattern of codon usage. The Z-curve as shown in Figure ?? measures codon usage patterns in a region of DNA. From the figure, one can observe that the read-through region matches the distribution before the regular stop codon. After the second stop however, the region matches regions found after regular stops.

Figure 4.15: Z-curve for Caki. Note that the codon usage in the read through region is similar to that in the region before the first stop codon.

Another suggestion offered in class was the possibility of ribosome slippage, where the ribosome skips some bases during translation. This might cause the ribosome to skip past a stop codon. This event occurs in bacterial and viral genomes, which have a greater pressure to keep their genomes small, and therefore can use this slipping technique to read a single transcript in each different reading frame. However, humans and flies are not under such extreme pressure to keep their genomes small. Additionally, we showed above that the excess we observe beyond the stop codon is frame specific to frame 0, suggesting that ribosome slipping is not responsible.

Cells are stochastic in general and most processes tolerate mistakes at low frequencies. The system isn’t perfect and stop codon leaks happen. However, the following evidence suggests that stop codon read-through is not random but instead subject to regulatory control:

- Perfect conservation of read-through stop codons is observed in 93% of cases, which is much higher than the 24% found in background.

- Increased conservation is observed upstream of the read-through stop codon.
Stop codon bias is observed. TGAC is the most frequent sequence found at the stop codon in read-through and the least frequent found at normal terminated stop codons. It is known to be a leaky stop codon. TAAA is found almost universally only in non-read-through instances.

Unusually high numbers of GCA repeats observed through read-through stop Conn.

Increased RNA secondary structure is observed following transcription suggesting evolutionarily conserved hairpins.

4.8.3 Excess Constraint

In most regions of the genome where we see conservation across species, we expect there to be at least some amount of synonymous substitution. These are “silent” nucleotide substitutions that modify a codon in such a way that the amino acid it encodes is unchanged. Looking across 29 mammals, the average codon site shows five synonymous substitutions. Given such a high average substitution rate, we do not expect to see perfect conservation across all regions that are conserved.

When we look at conserved regions, several stand out as being almost perfectly conserved with almost no synonymous substitutions. Below we look at some causes of excess constraint, how we can model it mathematically, and then see some real examples.

![Figure 4.16](image.png)

Figure 4.16: Many genomic regions, such as HOXB5, show more conservation than we would expect in normal conserved coding regions

4.8.4 Causes of Excess Constraint

The question is what evolutionary pressures cause certain regions to be so perfectly conserved? The following were all mentioned in class as possibilities:

- Could it be that there is a special structure of DNA shielding this area from mutation?
- Is there some special error correcting machinery that sits at this spot?
- Can the cell use the methylation state of the two copies of DNA as an error correcting mechanism? This mechanism would rely on the fact that the new copy of DNA is unmethylated, and therefore the DNA replication machinery could check the new copy against the old methylated copy.
- Maybe the next generation can’t survive if this region is mutated?

Another possible explanation is that selection is occurring to conserve specific codons. Some codons are more efficient than others: for example, higher abundant proteins that need rapid translation might select codons that give the most efficient translation rate, while other proteins might select for codons that give less efficient translation.
Still, these regions seem too perfectly conserved to be explained by codon usage alone. What else can explain excess constraint? There must be some degree of accuracy needed at the nucleotide level that keeps these sequences from diverging.

It could be that we are looking at the same region in two species that have only recently diverged or that there is a specific genetic mechanism protecting this area. However, it is more likely that so much conservation is a sign of protein coding regions that simultaneously encode other functional elements. For example, the HOXB5 gene shows obvious excess constraint, and there is evidence that the 5' end of the HOXB5 ORF encodes both protein and an RNA secondary structure.

Regions that encode more than one type of functional element are under overlapping selective pressures. There might be pressure in the protein coding space to keep the amino acid sequence corresponding to this region the same, combined with pressure from the RNA space to keep a nucleotide sequence that preserves the RNA's secondary structure. As a result of these two pressures to keep codons for the same amino acids and to produce the same RNA structure, the region is likely to show much less tolerance for any synonymous substitution patterns.

4.8.5 Modeling Excess Constraint

To better study regions of excess constraint, we develop mathematical models to systematically measure the amount of synonymous and nonsynonymous conservation of different regions. We will measure two rates: codon and nucleotide conservation.

For looking at substitutions at both the nucleotide and codon levels, we need to define the null model, where synonymous substitutions occur at some rate, and then define the alternative model that we can compare to the null model and get a measure of the amount of excess constraint.

To represent the null model, we can build rate matrices (4x4 in the nucleotide case and 64 * 64 for the codon case) that give the rates of substitutions between either codons or nucleotides for a unit time. We estimate the rates in the null model by looking at a ton of data and estimating the probabilities of each type of substitution. For the alternative model, we can define scaling factors in each region. These scaling factors measure how the rate of each type of substitution in our region of interest compares to that of the null model.

- \( \lambda_s \): the rate of synonymous substitutions
- \( \lambda_n \): the rate of nonsynonymous substitutions

For example, if \( \lambda_s = 0.5 \), then the rate of synonymous substitutions is half of what is expected from the null model in that region. We can then evaluate the statistical significance of the rate estimates we obtain, and find regions where the rate of substitution is much lower than expected.
Figure 4.18: We can measure $\lambda_s$ and $\lambda_n$, which give the rate of synonymous and nonsynonymous substitutions, respectively, in a given region.

Using a null model here helps us account for biases in alignment coverage of certain codons and also accounts for the possibility of codon degeneracy, in which case we would expect to see a much higher rate of substitutions. We will learn how to combine such models with phylogenetic methods when we talk about phylogenetic trees and evolution later on in the course.

Applying this model shows that the sequences in the first translated codons, cassette exons (exons that are present in one mRNA transcript but absent in an isoform of the transcript), and alternatively spliced regions have especially low rates of synonymous substitutions.

### 4.8.6 Examples of Excess Constraint

Examples of excess constraint have been found in the following cases:

- Most Hox genes show overlapping constraint regions. In particular, as mentioned above the first 50 amino acids of HOXB5 are almost completely conserved. In addition, HOXA2 shows overlapping regulatory modules. These two loci encode developmental enhancers, providing a mechanism to provide tissue specific expression.

- ADAR: the main regulator of mRNA editing, has a splice variant where a low synonymous substitution rate was found at a resolution of 9 codons.

- BRCA1: Hurst and Pal (2001) found a low rate of synonymous substitutions in certain regions of BRCA1, the main gene involved in breast cancer. They hypothesized that purifying selection is occurring in these regions. (This claim was refuted by Schmid and Yang (2008) who claim this phenomenon is the artifact of a sliding window analysis).

- THRA/NR1D1: these genes, also involved in breast cancer, are part of a dual coding region that codes for both genes and is highly conserved.

- SEPHS2: has a hairpin involved in selenocysteine recoding. Because this region must select codons to both conserve the protein’s amino acid sequence and the nucleotides to keep the same RNA secondary structure, it shows excess constraint.

### 4.8.7 Unusual miRNA Genes

The following four “surprises” were found when looking at specific miRNA genes:
• **Surprise 1**
  Both strands might be expressed and functional. For instance, in the miR-1ab-4 gene, expression of the sense and antisense strands are seen in distinct embryonic domains. Both strands score $> 0.95$ for miRNA prediction.

• **Surprise 2**
  Some miRNAs might have multiple 5' ends for a single miRNA arm, giving evidence for an imprecise start site. This could give rise to multiple mature products, each potentially with its own functional targets.

• **Surprise 3**
  High scoring miRNA* regions (the star arm is complementary to the actual miRNA sequence) are very highly expressed, giving rise to regions of the genome that are both highly expressed and contain functional elements.

• **Surprise 4**
  Both miR-10 and miR-10* have been shown to be very important Hox regulators, leading to the prediction that miRNAs could be “master Hox regulators”. Pages 10 and 11 of the first set of lecture 5 slides show the importance of miRNAs that form a network of regulation for different Hox genes.

### 4.9 Current Research Directions

### 4.10 Further Reading

### 4.11 Tools and Techniques

### 4.12 What Have We Learned?

Bibliography
CHAPTER
FIVE

GENOME ASSEMBLY AND WHOLE-GENOME ALIGNMENT

Melissa Gymrek, Liz Tsai

List of figures

Dynamic view of a changing gene ??
Mechanisms of chromosomal evolution ??
Moving further back in evolutionary time for Saccharomyces ??
Gene Correspondence for *S.cerevisiae* chromosomes and *K.waltii* scaffolds. ??
Gene interleaving ??
Gene correspondent ??
*S. cervisiae* chromosome VI gene correspondence ??

5.1 Introduction

In the previous chapter, we saw the importance of comparative genomics analysis for discovering functional elements. In “part IV” of this book, we will see how we can use comparative genomics for studying gene evolution across species and individuals. In both cases however, we assumed that we had access to complete and aligned genomes across multiple species.

In this chapter, we will study the challenges of genome assembly and whole-genome alignment that are the foundations of whole-genome comparative genomics methodologies. First, we will study the core algorithmic principles underlying many of the most popular genome assembly methods available today. Second, we will study the problem of whole-genome alignment, which requires understanding mechanisms of genome rearrangement, segmental duplication, and other translocations. The two problems of genome assembly and whole-genome alignment are similar in nature, and we close by discussing some of their parallel between them.
5.2 Genome Assembly I: Overlap-Layout-Consensus Approach

5.2.1 Finding overlapping reads
5.2.2 Merging reads into contigs
5.2.3 Laying out contig graph into scaffolds
5.2.4 Deriving consensus sequence
5.2.5 Dealing with sequencing errors

5.3 Genome Assembly II: String graph methods

5.3.1 String graph definition and construction
5.3.2 Bi-directed graphs and edge labels
5.3.3 Flows and graph consistency
5.3.4 Dealing with sequencing errors

5.4 Whole-Genome Alignment

5.4.1 Global, local, and 'glocal' alignment
5.4.2 Lagan: Chaining local alignments
5.4.3 Building Rearrangement graphs

5.5 Gene-based region alignment

An alternative way for aligning multiple genomes anchors genomic segments based on the genes that they contain, and uses the correspondence of genes to resolve corresponding regions in each pair of species. A nucleotide-level alignment is then constructed based on previously-described methods in each multiply-conserved region.

What makes it difficult is that not all regions have one-to-one correspondence and the sequence is not static genes undergo divergence, duplication, and losses and whole genomes undergo rearrangements. To help overcome these challenges, researchers look at the amino-acid similarity of gene pairs across genomes and the locations of genes within each genome.
Gene correspondence can be represented by a weighted bipartite graph with nodes representing genes with coordinates and edges representing weighted sequence similarity (Figure 5.2). Orthologous relationships are one-to-one matches and paralogous relationships are one-to-many or many-to-many matches. The graph is first simplified by eliminating spurious edges and then edges are selected based on available information such as blocks of conserved gene order and protein sequence similarity.

The Best Unambiguous Subgroups (BUS) algorithm can then be used to resolve the correspondence of genes and regions. BUS extends the concept of best-bidirectional hits and used iterative refinement with a increasing relative threshold. It uses the complete bipartite graph connectivity with integrated amino-acid similarity and gene order information.

In the example of a correctly resolved gene correspondence of *S.cerevisiae* with three other related species, more than 90% of the genes had a one-to-one correspondence and regions and protein families of rapid change were identified.
5.6 Mechanisms of Genome Evolution

Looking at a whole genome, we find that there are specific regions of rapid evolution. In *S. cerevisiae*, for example, 80% of ambiguities are found in 5% of the genome. Telomeres are repetitive DNA sequences at the end of chromosomes which protect the ends of the chromosomes from deterioration. Telomere regions are inherently unstable, tending to undergo rapid structural evolution, and the 80% of variation corresponds to 31 of the 32 telomeric regions. Gene families contained within these regions such as HXT, FLO, COS, PAU, and YRF show significant evolution in number, order, and orientation. Several novel and protein-coding sequences can be found in these regions. Since aside from the telomeric regions, very few genomic rearrangements are found in *S. cerevisiae*, regions of rapid change can be identified by protein family expansions in chromosome ends.

![Figure 5.4: Dynamic view of a changing gene](image)

Globally speaking, genes evolve at different rates: there are both fast and slow evolving genes. For example as illustrated in Figure ??, on one extreme, there is YBR184W in yeast which shows unusually low sequence conservation and exhibits numerous insertions and deletions across species. On the other extreme there is MatA2, which shows perfect amino acid and nucleotide conservation. Mutation rates often also vary by functional classification. For example, mitochondrial ribosomal proteins are less conserved than ribosomal proteins.

The fact that some genes evolve more slowly in one species versus another may be due to factors such as longer life cycles. Lack of evolutionary change in specific genes, however, suggests that there are additional biological functions which are responsible for the pressure to conserve the nucleotide sequence. Yeast can switch mating types by switching all their A and α genes and MatA2 is one of the four yeast mating-type genes (MatA2, Matα2, MatA1, Matα1). Its role could potentially be revealed by nucleotide conservation analysis.
Fast evolving genes can also be biologically meaningful. Mechanisms of rapid protein change include:

- Protein domain creation via stretches of Glutamine (Q) and Asparagine (N) and protein-protein interactions,
- Compensatory frame-shifts which enable the exploration of new reading frames and reading/creation of RNA editing signals,
- Stop codon variations and regulated read-through where gains enable rapid changes and losses may result in new diversity
- Inteins, which are segments of proteins that can remove themselves from a protein and then rejoin the remaining protein, gain from horizontal transfers of post-translationally self-splicing inteins.

We now look at differences in gene content across different species (*S.cerevisiae, S.paradoxus, S.mikatae, and S.bayanus*). A lot can be revealed about gene loss and conversion by observing the positions of paralogs across related species and observing the rates of change of the paralogs. There are 8-10 genes unique to each genome which are involved mostly with metabolism, regulation and silencing, and stress response. In addition, there are changes in gene dosage with both tandem and segment duplications. Protein family expansions are also present with 211 genes with ambiguous correspondence. All in all however, there are few novel genes in the different species.

### 5.6.1 Chromosomal Rearrangements

These are often mediated by specific mechanisms as illustrated for Saccharomyces in Figure 5.5.

Figure 5.5: Mechanisms of chromosomal evolution.

Translocations across dissimilar genes often occur across transposable genetic elements (Ty elements in yeast for example). Transposon locations are conserved with recent insertions appearing in old locations and long terminal repeat remnants found in other genomes. They are evolutionarily active however (for example with Ty elements in yeast being recent), and typically appear in only one genome. The evolutionary advantage of such locationally conserved transposons may lie in the possibility of mediating reversible arrangements. Inversions are often flanked by tRNA genes in opposite transcriptional orientation. This may suggest that they originate from recombination between tRNA genes.
5.7 Whole Genome Duplication

As you trace species further back in evolutionary time, you have the ability to ask different sets of questions. In class, the example used was *K. waltii*, which dates to about 95 millions years earlier than *S. cerevisiae* and 80 million years earlier than *S. bayanus*.

Looking at the dotplot of *S. cerevisiae* chromosomes and *K. waltii* scaffolds, a divergence was noted along the diagonal in the middle of the plot, whereas most pairs of conserved region exhibit a dot plot with a clear and straight diagonal. Viewing the segment at a higher magnification (Figure ??), it seems that *S. cerevisiae* sister fragments all map to corresponding *K. waltii* scaffolds.

Schematically (Figure ??) sister regions show gene interleaveing. In duplicate mapping of centromeres, sister regions can be recognized based on gene order. This observed gene interleaveing provides evidence of complete genome duplication.
Figure 5.8: Gene interleaving shown by sister regions in *K. waltii* and *S. cerevisae*

Figure 5.9: Merging reads.

Figure 5.10: Graph for overlapping reads once contig is assembled.

### 5.8 Additional figures

#### Bibliography


Figure 5.11: Shotgun sequencing.

Figure 5.12: S-LAGAN results.

Figure 5.13: S-LAGAN results for IGF locus.


Given above shotgun reads, construct graph:

Figure 5.14: S-LAGAN results for IGF locus.

Figure 5.15: String graph concept


CHAPTER SIX

BACTERIAL GENOMICS– MOLECULAR EVOLUTION AT THE LEVEL OF ECOSYSTEMS

Guest Lecture by Eric Alm
Scribed by TODO: missing @scribe: Who was this scribed by?

List of figures

TODO: missing @scribe: insert list of figures; find relevant images
6.1 Introduction

The microbiome as an ecosystem inside the human body. Data source: the Human Microbiome Project (HMP) generated much of the data referred to in this lecture. Relevant numbers:

- 10 fold more bacterial cells in the human body than human cells \(O(1e14)\)
- 100x more distinct genes than in the human genome

Goal: model the microbial ecosystem. This problem can be divided into several parts as follows:

1. Survey: survey by high throughput sequencing of 16S ribosome, barcodes for species: who is present and where?
2. Rules: Analysis of rules/modules/motifs/clusters in ecosystem occupancy patterns as a function of space and time
3. Rules and motifs in ODE model (species as a function of time)
4. Incorporate spatial structure in PDE model (time and space)

6.2 Further Details on ODE Modeling

As an example of a model in step 3, consider a simple coupled ODE model for the population of an e.coli species within a niche in the microbiome:

\[ N_i = e_{coli} \]

The differential is proportional to abundance of species times some other function of the abundance of other species and a lumped parameter \(E\) for environmental factors.

- \(a_{ij}\) = species-species interaction term
- \(r_i\) = intrinsic growth rate
- \(K_j(E(t))\) = carrying capacity for species \(j\)

6.3 Further Details on Data Generation Process

- Extract DNA from body site
- Amplify 16S rDNA to get prokaryotic DNA marker
- Sequence this to get a large number of molecular barcodes (don’t talk about species)
- instead of species, classify (cluster) into OTU (operational taxonomic unit)
- get out vector of abundances of the different species (normalized counts of OTUs)

6.4 Species-Environment Associations

Applications in disease diagnosis: (look at bacteria to diagnose human physiology) and natural biosensors.

Current unsupervised learning approach:

- PCA (principal components analysis) on the microbial abundance vector space
- cluster in the reduced dimensional space

Limitations of this approach:

- the resulting clusters correspond reasonably well to a few different environments, but have trouble distinguishing MANY factors. Not getting data out that is as rich as the data itself
• PCA always gives the same plot. Clustering always gives the same clusters! No way to incorporate external information. This is a major limitation.

SLIME (supervised learning in microbial ecology) is a supervised learning approach which overcomes the above limitations. Here the axes also become biologically meaningful, unlike in PCA. **Challenges:**

• large inter-patient variation
• low abundance discriminative taxa

**Techniques:**

• Random forests
• Support vector machines

General rule = avoid unsupervised learning when know categories already

**HMP body part study using SLIME:**

• 18 body sites on 250 individuals 16s rDNA sequences
• high discriminative accuracy between closely spaced sites

• consider superyingival plaque vs. subgingival plaque (above or below gumline): 73 percent cross-validated accuracy at predicting differences; key bacteria that distinguish are: strict anaerobic vs. facultative anaerobic species (facultative = can tolerate oxygen). This makes sense more oxygen above the gumline vs. below

• Vaginal sites: dominated by lactobacillus, except that some patients have much fewer lactobacillus and higher abundance of other species interestingly, these were perfectly healthy patients but these bacteria were the characteristic ones of bacterial vaginosis. Therefore, the bacteria population could be a normal state of healthy people who, however, have a greater propensity to get bacterial vaginosis for OTHER reasons. The bacteria may not be the causative factor.

**6.4.1 Pediatric Inflammatory Bowel Disease: collaboration w/ Childrens hospital**

(Athos Bousvaros)

• goal = diagnose without colonoscopy

• Random forest classifier on the species with known categories sick vs. non-sick patients with large additional sources of inter-patient variability

• Note: Shannon diversity of an ecosystem (assuming this is the same as the Shannon entropy measure on the species occupancies). As patients become more diseased, ecosystem diversity decreases. Could be the ecosystem as a whole that is controlling disease status, not one particular species!

• effect of sampling depth
• What about unsupervised learning approaches?

**6.4.2 Topic Models: Basic Concepts**

• plaques divide into contributions from sub-gingival, supra-gingival and rest of mouth = basic idea is assume underlying sub-populations or sources of a sample

• infer probability of a sample from a given topic model of that sample

• run maximum likelihood methods to find best topic model for a sample set
• Strongest prediction: one individual female has a different topic model turns out to come from chloroplasts (some sort of face product?)

Bioremediation study at Oak Ridge NL: need to clean up chromium
• identify taxa responsive to nitrate
• nitrate inhibits Cr bioremediation
• infer environmental conditions from DNA
• want to identify shift in microbial community structure as fxn of nitrate concentration
• available data gives rise to tricky prediction problem
  – Desulfovibrio are predictive of nitrate level. These are the only ones.
  – Strongly suggests desulfovibrio species are playing a key role in the bioremediation process.
  – Can use DNA sequences give a very good regressor to nitrate concentration

6.5 Species-species interactions
co-occurrence networks: find species-species correlations across different environment
• currently: lots of strong correlations but the involved species appear different between different techniques
• need to normalize data because of experimental sample-sample variability: this totally changes the nature of the analysis problem (compositional effects)
• also see anti-correlations: can draw in a colored graph. Item Graduate student wanted to prove existence of compositional effects, so scrambled the data (lost sample-identity information) and got the SAME correlations. After all Pearson said dont use his correlation coefficient for rational functions of the variables youre interested in, such as frequency:

\[
 f_i = \frac{\text{abundance}_i}{\text{sum}_i(\text{abundance}_i)}
\]

The basic idea is that you’re seeing spurious positive and negative correlations when you deal with percentages. They tell about overall abundance of species but not about correlated fluctuations in bacterial abundances.

solution: use log frequency ratio matrix \( y_{ij} = \log \frac{f_i}{f_j} \) and compute the variance of each element in this matrix. Overall abundances cancel out in the ratio. Can relate this variance matrix to the species-species covariance matrix which is what we were really interested in! (this is assuming many components with sparse correlations)

What we understand now is that with the pearson correlation there is a low effective number of species. Only with high #s of species do we get something meaningful. But with sparse correlation inference method, can infer the right correlations even at low effective number of species. Interesting observation: something as simple as normalizing your data and taking a correlation is an open problem in this field.

6.6 Time series models: moving to ODEs

Experiment: inject new bacteria and see how population dynamics is affected
• HuGE network: effects of environmental factors (sleep, happiness) on gut ecosystem
• Horizontal gene transfer (HGT) in the human microbiome: maybe the ODEs totally break down due to gene exchange
– looking only at the most recent gene transfers
– want to identify 100% identical DNAs in unrelated species
– there is a significant amount of identical DNA at all levels of phylogenetic distance
– inside human genomes there is MORE hgt: almost all hgt in genbank is due to bacteria isolated from human bodies, and much is happening in the same site of the body (in different people but same SITES) outside the body human food also has a lot of DNA swapping (human food to gut transfer)
– highest levels of swapping are from the same micro-environment
– also see exchange from commensal bacteria to pathogens and pathogen to pathogen innovations in pathogens are rapidly shared

6.7 Current Research Directions
6.8 Further Reading
6.9 Tools and Techniques
6.10 What Have We Learned?

Bibliography
Part II

Coding and Non-Coding Genes
CHAPTER SEVEN

HIDDEN MARKOV MODELS I

Gleb Kuznetsov, Sheida Nabavi (Sep 28, 2010)
Elham Azizi (Sep 29, 2009)

List of figures

Modeling biological sequences ??
Motivating example: the dishonest casino ??
Potential DNA sources: viral injection vs. normal production ??
Running the model: probability of a sequence (I) ??
Running the model: probability of a sequence (II) ??
Comparing the two models ??
Partial runs and die switching ??
Rolling a die with repetitions of a number ??
HMMS as a generative model ??
Probability of seq. path if all promoter ??
Probability of seq. path if all background ??
Probability of seq. path sequence if mixed ??
The six algorithmic settings for HMMS ??
Calculate total probability recursively ??
The forward algorithm ??
7.1 Introduction

Hidden Markov Models (HMMs) are some of the most widely used methods in computational biology. They allow us to investigate questions such as uncovering the underlying model behind certain DNA sequences. By representing data in rich probabilistic ways, we can ascribe meaning to sequences and make progress in endeavors including, but not limited to, Gene Finding. This lecture is the first of two on HMMs. It covers Evaluation and Parsing. The next lecture on HMMs will cover Posterior Decoding and Learning with an eventual lead into the Expectation Maximization (EM) algorithm. We will eventually cover both supervised and unsupervised learning. In supervised learning we have training data available that labels sequences with particular models. In unsupervised learning we do not have labels so we must seek to partition the data into discrete categories based on discovered probabilistic similarities. We find many parallels between HMMs and sequence alignment. Dynamic programming lies at the core of both methods, as solutions to problems can be viewed as being composed of optimal solutions to sub-problems.

7.2 Modeling

7.2.1 We have a new sequence of DNA, now what?

1. Align it with database we know labels for (database search)
2. Align to unknown things (assemble/clustering)
3. Classify as protein, motif, etc. by features such as GC content
4. Stare at it! For example:
   - Look for nonstandard nucleotide compositions
   - Look for k-mer frequencies that are associated with protein coding regions; recurrent data; high GC content; etc.
   - Genomics rule #1: Look at your data! visualize and look for patterns, then develop machine learning to develop into probabilistic models. For example by looking at a number of quadruples we decide to color code them to see where they most frequently occur.
5. Model it: By making hypothesis and building a generative model to describe it and use that model to find sequences of similar type. We’re not looking for sequences that necessarily have common strings but we’re interested in similar properties. We actually don’t know how to model whole genomes, but we can model small aspects of genomes. The task requires understanding all the properties of genome regions and computationally building generative models to represent hypotheses. For a given sequence, we want to annotate regions whether they are introns, exons, intergenic, promoter, etc regions.

![Figure 7.1: Modeling biological sequences](image)

Building this framework will give us the ability to:
- Generate (emit) sequences of similar type according to the generative model
- Recognize the hidden state that has most likely generated the observation
- Learn (train) large datasets and apply to both previously labeled data (supervised learning) and unlabeled data (unsupervised learning).

In this lecture we discuss algorithms for emission and recognition.
7.2.2 Why probabilistic sequence modeling?

- Biological data is noisy
- Probability provides a calculus for manipulating models
- Not limited to yes/no answers can provide degrees of belief
- Many common computational tools based on probabilistic models
- Our tools: Markov Chains and HMM

7.3 Motivating Example: The Dishonest Casino

7.3.1 The Scenario

Imagine the following scenario: You enter a casino that offers a dice-rolling game. You bet $1 and then you and a dealer (that handsome individual in the photo above) both roll a die. If you roll a higher number you win $2. Now there’s a twist to this seemingly simple game. You are aware that the casino has two types of die:

1. Fair die: \( P(1) = P(2) = P(3) = P(4) = P(5) = P(6) = \frac{1}{6} \)
2. Loaded die: \( P(1) = P(2) = P(3) = P(4) = P(5) = \frac{1}{10} \) and \( P(6) = \frac{1}{2} \)

The dealer can switch between these two die at any time without you knowing it. The only information that you have are the rolls that you observe. We can represent the state of the casino die with a simple Markov model:

![Markov Model Diagram](Figure 7.2: State of a casino die represented by a hidden Markov model)

The model shows the two possible states, their emissions, and probabilities for transition between them. The transition probabilities are educated guesses at best. We assume that switching between the states doesn’t happen too frequently, hence the .95 chance of staying in the same state with every roll.

7.3.2 Staying in touch with biology: An analogy

For comparison, the figure below gives a similar model for a situation in biology where a sequence of DNA has two potential sources: injection by a virus versus normal production by the organism itself:

Given this model as a hypothesis, we would observe the frequencies of CpG to give us clues as to the source of the sequence in question.
7.3.3 Running the Model

Let's look at a particular sequence of rolls and pick for the underlying model the two extreme cases: One in which the dealer is always using a fair die, and the other in which the dealer is always using a loaded die. We run the model on each to understand the implications. We build up to introducing HMMs by first trying to understand the likelihood calculations from the standpoint of fundamental probability principles of independence and conditioning.

In the first case, where we assume the dealer is always using a fair die, we calculate the probability as shown in the figure above. The product term has three components: - the probability of starting with the fair die model $1/6^{10}$ - the probability of the roll given the fair model (6 possibilities with equal chance) $0.95^9$ - the transition probabilities that keep us always in the same state

The opposite extreme, where the dealer always uses a loaded die, has a similar calculation, except that we note a difference in the emission component. This time 8 of the 10 rolls carry a probability of $1/10$ based on the assumption of a loaded die underlying model, while the two rolls of 6 have a probability of occurring. Again we multiply all of these probabilities together according to principles of independence and conditioning.

Finally, in order to make sense of the likelihoods, we compare them by calculating a likelihood ratio. We find that it is significantly more likely for the sequence to result from a fair die. Now we step back and ask, does this make sense? Well, sure. Two occurrences of rolling 6 in ten rolls doesn't seem out of the
ordinary so our intuition matches the results of running the model.

7.3.4 Adding Complexity

Now imagine the more complex, and interesting, case where the dealer switches the die at some point during the sequence. We make a guess at an underlying model based on this premise:

Again, we can calculate the likelihood using fundamental probabilistic methods. Now, it begins to dawn on us that we will need more rigorous techniques for inferring the underlying model. In the above cases we more-or-less just guessed at the model, but what we want is a way to systematically derive likely models. Let's formalize the models introduced thus far as we continue toward understanding HMM-related techniques.

7.4 Formalizing Markov Chains and HMMS

7.4.1 Markov Chains

A Markov Chain reduces a problem space to a finite set of states and the transition probabilities between them. At every time step, we observe the state we are in and simulate a transition, independent of how we
got that state. More formally, a Markov Chain is a memory-less series of states $Q$ with transitions. In order to represent it, we require:

- Transition probabilities through a matrix $A$ whose elements give the probabilities of transition from state $i$ to state $j$.
- Initial state (vector probabilities): $p$

The key property of Markov Chains is that they are memory-less, i.e. the probability of each symbol depends only on the previous state. So we can immediately define a probability for the next state, given the current state:

$$P(x_i|x_{i-1},...,x_1) = P(x_i|x_{i-1})$$

Therefore, the columns of $A$ have to sum up to 1. In this way, the probability of the sequence can be decomposed into:

$$P(x) = P(x_L,x_{L-1},...,x_1) = P(x_L|x_{L-1})P(x_{L-1}|x_{L-2})...P(x_2|x_1)P(x_1)$$

$P(x1)$ can also be calculated from the transition probabilities: If we multiply the initial vector of probabilities at time $t = 0$ by the transition matrix, we get the probabilities of states at time $t = 1$ and therefore at time time $t = n$.

### 7.4.2 Hidden Markov Models

Hidden Markov Models are used as a representation of a problem space in which observations come about as a result of states of a system that we are unable to observe directly. These observations, or emissions, result from a particular state based on a set of probabilities. Thus HMMs are Markov Models where the states are hidden from the observer and instead we have observations generated by emission probabilities associated with the each state.

Summarizing the parameters:

- Series of states $Q$
- Transition matrix $A$: For each $s, t$ in $Q$ the transition probability is: $a_{st}P(x_i = t|x_{i-1} = s)$
- Initial state probabilities $p$.
- $V$: set of observation symbols, for example A, T, C, G, or 20 amino-acids or utterances in human language.
- $E$: matrix of emission probabilities: $e_{sk}P(v_k|attimet|q_t = s)$
The key property of memory-less-ness is inherited from Markov Models: the emissions and transitions are only dependent on the current state and not on the past.

HMMs are a great tool for gene sequence analysis, because we can look at a sequence of DNA as being emitted by a mixture of models. These may include introns, exons, transcript factors, etc. While we may have some sample data that matches models to DNA sequences, in the case that we start fresh with a new piece of DNA, we can use HMMs to ascribe some potential models to the DNA in question.

### 7.5 Back to Biology

We will introduce an example and think about it a bit, before finally describing the HMM techniques that solve the problems that arise in such a first-attempt/native analysis.

Imagine the following scenario: we are trying to find GC rich regions by modeling nucleotide sequences drawn from two different distributions: background and promoter. Background regions have uniform distribution of 0.25 for each of ATGC. Promoter regions have probabilities: A: .15, T: 0.13, G: 0.30, C: 0.42. Given one nucleotide observed, do not know which region it is originated from, but both can emit at different probabilities. We have also learned the initial state probabilities based in steady state probabilities. By looking at a sequence we want to identify which regions originate from a background distribution (B) and which regions are from a promoter model (P). It was noted again that Markov chains are absolutely memory-less, this means that for example if you have lost in a Casino in the last 7 days, it wont mean that you would most probably win today. This is also true in the example of rolling a die where you might have repetitions of a number.

![Figure 7.8: Rolling a die with repetitions of a number](image)

![Figure 7.9: HMMs as a generative model](image)

We are given the transition and emission probabilities based on relevant abundance and average length of regions where X = vector of observable emissions: A,T,G,C; \( \pi = vector \) of paths (e.g. BPPBP); \( \pi^* \) = maximum likelihood of generating that path. In our interpretation of sequence, the max likelihood path will be found by incorporating all emissions transition probabilities (by dynamic programming).

HMMs are generative models, in a way that it gives the probability of emission given a state (with Bayes Rule), essentially telling you how likely it is to generate those sequences. So we can always run a generative
model for transition between states and start anywhere. In Markov Chains the next state will give different outcomes with different probabilities. No matter which the next state is, at that next state, the next symbol will still come out with different probabilities. HMMs are similar: You can pick an initial state based on initial probability vector. In the example above, you pick B since there is more background than promoter. Then draw an emission from the P(X — B). Since all are 0.25, you can pick any symbol, for example G. Given this emission, the probability of the next transition state does not change. So we have transition to B again is with probability 0.85 and to P with 0.15 so we go with B and so on.

We can compute the probability of one such generation by multiplying the probabilities that the model makes exactly the choices we assumed. For the example above we have the probability of three different paths computed as below:

Path 1:

Path 2:

Path 3:

And therefore we score the sequence in that way and compare scores of different paths.
Now the question is what path is most likely to generate the given sequence?

By brute force search we can look at all paths, try all possibilities, calculate their joint probability $P(x, \pi)$. The sum of probabilities of all the alternatives is 1. For example if all states are promoters, $P(x, \pi) = 9.3E-7$. If all emissions are Gs, $P(x, \pi) = 4.9E-6$. If we have a mixture, $P(x, \pi) = 6.7E-7$; which is small because a lot of penalty is paid for transition between B’s and P’s which is exponential in length of sequence.

If you observe more G’s, it is more likely in the promoter region and A&Ts are in the background. This score can be interpreted as additive scores in logs and the Viterbi path is the most likely path of joint probabilities.

### 7.6 Algorithmic Settings for HMMs

<table>
<thead>
<tr>
<th>The six algorithmic settings for HMMs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One path</strong></td>
</tr>
<tr>
<td>1. Scoring $x$, one path</td>
</tr>
<tr>
<td>$P(x, \pi)$</td>
</tr>
<tr>
<td>Prob of a path, emissions</td>
</tr>
<tr>
<td>2. Viterbi decoding</td>
</tr>
<tr>
<td>$\pi^* = \text{argmax}_\pi P(x, \pi)$</td>
</tr>
<tr>
<td>Most likely path</td>
</tr>
<tr>
<td>3. Supervised learning, given $\pi$</td>
</tr>
<tr>
<td>$\Lambda^* = \text{argmax}_\lambda P(x, \pi</td>
</tr>
<tr>
<td>5. Unsupervised learning</td>
</tr>
<tr>
<td>$\Lambda^* = \text{argmax}_\lambda \Sigma_p P(x, \pi</td>
</tr>
<tr>
<td>6. Unsupervised learning</td>
</tr>
<tr>
<td>$\Lambda^* = \text{argmax}_\lambda \Sigma_p P(x, \pi</td>
</tr>
<tr>
<td><strong>All paths</strong></td>
</tr>
<tr>
<td>2. Scoring $x$, all paths</td>
</tr>
<tr>
<td>$P(x) = \Sigma_{\pi} P(x, \pi)$</td>
</tr>
<tr>
<td>Prob of emissions, over all paths</td>
</tr>
<tr>
<td>4. Posterior decoding</td>
</tr>
<tr>
<td>$\pi^\wedge = { \pi_i</td>
</tr>
<tr>
<td>Path containing the most likely state at any time point.</td>
</tr>
<tr>
<td>5. Unsupervised learning</td>
</tr>
<tr>
<td>$\Lambda^* = \text{argmax}_\lambda \Sigma_p P(x, \pi</td>
</tr>
<tr>
<td>6. Unsupervised learning</td>
</tr>
<tr>
<td>$\Lambda^* = \text{argmax}_\lambda \Sigma_p P(x, \pi</td>
</tr>
<tr>
<td>Baum-Welch training, over all paths</td>
</tr>
</tbody>
</table>

Figure 7.13: The six algorithmic settings for HMMs

We use HMMs for three types of operation: scoring, decoding and learning. We talk about scoring and decoding in this lecture. These operations can happen for a single path or all possible paths. For the single path operations, our focus is on discovering the path with maximum probability, the most likely path. However in all paths operations we are interested in a sequence of observations or emissions regardless of its corresponding paths.

#### 7.6.1 Scoring

Scoring for a single path is asking for the joint likelihood of observing a particular sequence and having a particular path: $P(x, \pi)$. The single path calculation is the likelihood of observing the given sequence over a particular path. For the all paths version, we do all operations on all possible paths and states and score over all paths $\pi$: $P(x) = \Sigma_{\pi} P(x, \pi)$. In this case the score is calculated for just a given sequence of observations or emissions regardless of the paths. We are using this score when we are interested in knowing the likelihood of particular sequence for a given HMM.
7.6.2 Decoding

Decoding answers the question: Given some observed sequence, what path gives us the maximum likelihood of observing this sequence? We can imagine a brute force approach where we calculate the joint probabilities of a given emission and all possible paths and then pick the path with the maximum joint probability. The problem is there are too many paths and using such a brute force search for the maximum likelihood among all the paths is very time consuming and is not practical. To solve this problem dynamic programming can be used. Let us restate the problem in the dynamic programming way of thinking.

Through decoding we would like to find out the most likely sequence of states based on the observation. For the decoding operation, the model parameters $e_{is}$ (the emission probabilities given their states) and $a_{ij}$, (the state transition probabilities) are given. Also, the sequence of emissions $X$ is given. The goal is to find the sequence of hidden states, $\pi$, which maximizes the joint probability of a given emission and its path $P(x, \pi)$.

Given the emitted sequence $X$ we can evaluate any path through hidden states. However we are looking for the best path. The best path through a given state must contain within it the following:

- Best path to previous state
- Best transition from previous state to this state
- Best path to the end state

Therefore the best path can be obtained based on the best path of the previous states. The Viterbi algorithm, a dynamic programming algorithm, is commonly used to obtain the best path. Assuming $V_k(i)$, which is the probability of the most likely path through state $k$ at position (or time instance) $i$ in the path (which can be shown as $\pi_i = k$) is known. Then we can compute this probability at time $i + 1$, $V_k(i + 1)$, as a function of $\max_k V_k(i)$ as follows. Basically we are reusing the computation we have already done. $V_k(i + 1) = e_k(x_{i+1}) \times \max_{a_{ij}} V_j(i)$ Now we want to find the most likely path. The most likely path, $\pi^*$, is the path that maximizes the total joint probability $P(x, \pi)$.

In the examples about fair die and loaded lie we saw how we can obtain $P(x, \pi)$ for a particular path and observation. We saw that $P(x, \pi)$ can be computed by multiplying starting transition probability $\theta$ with multiplications of emissions probabilities and transition probabilities at each time point $\lambda$, as expressed below.

The most probable path $\pi^*$ or the maximum $P(x, \pi)$ can be found recursively. Assuming we know $V_j(i - 1)$, the score of the maximum path up to time $i$, now we need to increase the computation for the next time step. The new maximum score path for each state depends on the maximum score of the previous states, the penalty of the transition to the new states (transition probability) and the emission probability. In other words, the new maximum score for a particular state at time $i$ is the one that maximizes the transition of all possible previous states to that particular state (the penalty of transition) multiplied by their maximum previous scores multiplied by emission probability at the current time as expressed as follows.

All sequences have to start in state 0 (the begin state). By keeping pointers backwards the actual state sequence can be found by backtracking. The solution of this Dynamic Programming problem is summarized in the following slide. It is very similar to the alignment.

We are filling the above matrix from left to right and trace back. Each point in the matrix has K operation and there are KN points so the required computation time is $O(K2N)$ and the required space is $O(KN)$ to remember the pointers. Noting that the running time has reduced from exponential to polynomial.

7.6.3 Evaluation

Evaluation is about answering the question: How well does our model of the world capture the actual world? Given a sequence $x$, many paths can generate this sequence. The question is how likely is the sequence, given the entire model? In other words, is this a good model or how well the model does capture the exact characteristics of a particular sequence? This kind of operation of HMMS is evaluation. With evaluation we can compare the models. So similar to decoding the model parameters $e_i(.)s$ (the emission probabilities given their states) and $a_{ij}$, (the state transition probabilities) are given. Also, the sequence of emissions or observation $X$ is given. The goal is to find the total probability of a particular sequence (given the model). In other words, we would like to find $P(x|M)$ where $M$ is the model. In this case we are not interested in a
particular path. What we want to know is, what is the probability that \( X \) was generated by the given model (using any path)? Or we want to obtain the following probability:

\[
P(x) = \Sigma \pi P(x, \pi)
\]

The challenge of obtaining this probability is there are too many paths (exponential number of paths). Each path has an associated probability. Some paths are more likely, under unlikely, and we need to sum them up. One solution is using just the Viterbi path and ignoring the others. We already have the tool to obtain the Viterbi path. But its probability is very small; it is just a small fraction of the mass probability of all the possible paths. It is a good approximation only if it has high density. In other cases it will give us inaccurate approximation. Alternative solution is calculating the exact sum iteratively by using dynamic programming. The Forward Algorithm to obtain the forward probability \( f(i) \) can be derived as follow.

\[
fl(i) = P(x1xi, \pi = l)
\]

\[
= Sp1\pi - 1P(x1xi - 1, p1, \pi - 2, \pi - 1, \pi = l)el(xi)
\]

\[
= SkSp1\pi - 2P(x1xi - 1, p1, \pi - 2, \pi - 1 = k)aklel(xi)
\]

\[
= Skfk(i - 1)aklel(xi)
\]

\[
= el(xi)Skfk(i - 1)akl
\]

Therefore the total probability \( P(x) \) can be calculated recursively as shown below.

The forward algorithm is shown in the following slide.

It can be seen that the Forward algorithm is very similar to the Viterbi algorithm. In the Forward algorithm summation is used instead of maximization. Here we can reuse computations of the previous problem including penalty of emissions, penalty of transitions and sums of previous states. The drawback of this algorithm is that in practice taking the sum of logs is difficult; therefore approximations and scaling of probabilities are used instead.
7.7 Current Research Directions

7.8 Further Reading

7.9 Tools and Techniques

7.10 What Have We Learned?

Bibliography
CHAPTER

EIGHT

HIDDEN MARKOV MODELS II - POSTERIOR DECODING AND LEARNING

Amer Fejzic (2010)
Elham Azizi (2009)

List of Figures

TODO: missing @scribe: insert list of figures
8.1 Introduction

In the last lecture we got familiar with the concept of discrete-time Markov chains and Hidden Markov Models (HMMs). A Markov chain is a discrete random process that abides by the Markov property, that the probability of the next state depends only on the current state and not the past. The Markov chain models how a state changes from step to step using transition probabilities. Therefore, a Markov Model (MM) is fully defined as:

- \( \pi_i \in Q \), the state at the \( i^{th} \) step in a sequence of finite states \( Q \) of length \( N \) that can hold a value from a finite alphabet \( \Sigma \) of length \( K \)
- \( a_{jk} \in A \), the state transition probability of moving from state \( j \) to state \( k \), \( P(\pi_i = k | \pi_{i-1} = j) \), for each \( j,k \) in \( Q \)
- \( a_{0j} \in P \), the probability that the initial state will be \( j \)

\( A \) is the stochastic matrix. The probabilities of the leaving the state must sum to one, \( \sum_k a_{jk} = 1, \forall j \). The value a state can hold for example in the Dishonest Casino problem would be \( \pi_i = \{ \text{"fair die"}, \text{"loaded die"} \} \), or \( \pi_i = \{ \text{"background"}, \text{"promoter region"} \} \) for the GC-rich finding problem. The state transition probabilities describe the chance of a state “fair die” to change to “loaded die” versus staying “fair die”. This is were one models the likelihood of the switching to occur. Here the states are completely “visible”, as if there was a marker on the fair or loaded die and you can see it, or the GC-rich areas were already labeled from the background. Realistically, one only sees the number that the die shows or the A, T, G, C nucleotide sequence one observes from a gene. As a result, the value of the state is not seen and is therefore considered “hidden”. This led us to the Hidden Markov Models, a further generalization of the Markov Model.

An HMM adds the mechanics on top of the MM to model the discrete random process of the “observations” that are coupled with the states. For an \( N \) long sequence, there are now also \( N \) number of observations. To model this, each state emits a character from a given alphabet with a certain probability and the emitted characters are observed. Therefore, the two additional descriptors of an HMM on top of the MM as listed above is:

- \( x_i \in X \), the emission at the \( i^{th} \) step in a sequence of finite characters \( X \) of length \( N \) that can hold a character from a finite set of observation symbols \( v_l \in V \)
- \( e_{k}(v_l) \in E \), the emission probability of emitting character \( v_l \) when the state is \( k \), \( P(x_i = v_l | \pi_i = k) \)

The emission probabilities must also sum to one, \( \sum_l e_{k}(v_l) = 1, \forall k \). There are three things to consider about the HMM:

- \( a_{jk}, e_{k}(v_l) \), and \( a_{0j} \) that model the discrete random process
- \( \pi_i \), the sequence of hidden states
- \( x_i \), the sequence of emissions that are observed

Previously we have showed when given the full HMM \((Q, A, X, E, P)\) what is the likelihood that the discrete random process produced the given hidden states and emissions.

\[
P(x_1, \ldots, x_N, \pi_1, \ldots, \pi_N) = a_{0\pi_1} \prod_i e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}}
\]  

(8.1)

This is the total joint probability \( P(x, \pi) \). Usually the hidden states are not given and so we then asked the decoding problem, given the partial HMM \((A, X, E, P)\), infer the hidden states sequence \( Q \) that maximizes the total joint probability. This lead us to the Viterbi decoding algorithm. The algorithm leveraged the use of the principle of optimality and runs in \( O(K^2N) \) time and \( O(KN) \) space, where \( K \) is the number of states.

This lecture will discuss Posterior Decoding, which again will infer the hidden states sequence \( X \) that maximizes some other metric. It finds the most likely state at every position over all possible paths. The
algorithm relies on both the **forward** and **backward** algorithm. The forward algorithm computes the total probability of a given emission sequence being generated by a particular HMM over all possible state paths that could have generated it, and was discussed in last lecture. The backward algorithm is closely related. Afterwards we will show how to encode “memory” in a Markov chain such as the search for di-nucleotide CpG islands. Then we will discuss learning by means of Maximum Likelihood, EM, and Viterbi learning and we’ll discuss different learning frameworks for estimating parameters where we may be given a labeled dataset or a genome completely unlabeled. From there we will infer annotations and parameters for learning.

However the question is what will constrain the model in the unlabelled case? And we will see that it is the framework/structure we are given apriori. That is, the data itself is going to tell where the states are because it contains patterns embedded in it.

The example we are going to discuss is high GC and low GC regions and we wish to fit some states to high GC and some to low GC regions.

### 8.2 Posterior Decoding

#### 8.2.1 Forward Algorithm

This algorithm will allow us to compute \( P(x_1, \ldots, x_N) \), the probability of a sequence of emissions \( X \) given the parameters of the full HMM \((Q, A, X, E, P)\). Let’s rewrite this probability as the sum over all possible ways of generating the last emission in the sequence, \( x_N \):

\[
P(x_1, \ldots, x_N) = \sum_l P(x_1, \ldots, x_N, \pi_N = l) \tag{8.2}
\]

Define the quantity \( f_l(i) \) as the joint probability of generating \( x_1, \ldots, x_i \) (the first \( i \) emissions of the sequence) from the HMM and ending up in state \( l \) at step \( i \):

\[
f_l(i) = P(x_1, \ldots, x_i, \pi_i = l) \tag{8.3}
\]

Now rewrite \( P(x_1, \ldots, x_N) \) in terms of \( f_l(i) \):

\[
P(x_1, \ldots, x_N) = \sum_l f_l(n) \tag{8.4}
\]

In last lecture, the recursion formula of \( f_l(i) \) was derived:

\[
f_l(i) = e_l(x_i) \sum_k f_k(i - 1) a_{kl} \tag{8.5}
\]

We can now compute \( f_l(i) \) based on \( f_l(i - 1) \) summing over all probability in previous state and because these emissions are in fact identical in the sum we can extract the emissions. The forward algorithm is shown in Figure ??.

#### 8.2.2 Application

We have computed the probability of a sequence summed over all paths. But why would we ever want \( P(X) \)? Suppose we’re given 2 HMMs that can model your genomes. In the first one Promoters are modeled so that only Cs and Gs matter while in the second one, the CpGs (where the p denoted that they’re on the same strand) matter. You can calculate the total probability of explaining the data given the model while you don’t know where the promoters are. Given all possible parses of the sequence you can still compute the probability and compare the two HMMs.

Markov Chains and HMMs are memoryless, but there is trick to add memory by increasing the number of states, (actually, square the num of states). Similarly, we can combine two models of:

- ‘+’ model: from CpG islands
- ‘-’ model: from non-islands
6.047/6.878 Lecture 7: Hidden Markov Models II - Posterio Decoding and Learning

Figure 8.1: The Forward Algorithm

- **Input:** $x = x_1 \ldots x_N$
- **Initialization:**
  - $f_0(0) = 1$, $f_0(1) = 0$, for all $k > 0$
- **Iteration:**
  - $f_k(i) = e_{\pi}(x_i) \times \sum_{j \in \mathbb{S}} a_{ij} f_{k-1}(j)$
- **Termination:**
  - $\delta(x, \pi^*) = \max_k f_k(N)$

**In practice:**
- Sum of log scores is difficult
  - approximate $\propto \log(1+e^{-q})$
  - scaling of probabilities

**Running time and space:**
- Time: $O(KN)$
- Space: $O(K)$

Figure 8.2: HMM for CpG Islands

- **HMM for CpG islands**
  - Build a single model that combines two such Markov chains:
    - ‘+’ states: $A, C, G, T_+$
      - Emit symbols: $A, C, G, T$ in CpG islands
    - ‘−’ states: $A, C, G, T_-$
      - Emit symbols: $A, C, G, T$ in non-islands
  - Emission probabilities distinct for the ‘+’ and the ‘−’ states
  - Infer most likely set of states, giving rise to observed emissions
  - ‘Paint’ the sequence with ‘+’ and ‘−’ states

**Why we need so many states...**
In our simple GC-content example, we only had 2 states (P|B)
Why do we need 8 states here: 4 CpG+ / 4 CpG-?
- Encode ‘memory’ of previous state: nucleotide transitions
To give 4 ‘+’ states and 4 ‘-’ states which have different emission probabilities is shown in Figure ??.

The probability $P(\pi_i = k|x_i = G)$ is equal to the prior of being in state $k$ at position $i$ times the most likely emission. Now what if we know the entire sequence? Having observed the entire sequence, we want to know what is the probability by summing all possible paths which is what posterior decoding does. To do this efficiently, we implement the forward and backward algorithm Figure ?? and we already know how to compute the first part which is forward algorithm.

### 8.2.3 Backward Algorithm

Now to compute the backward algorithm, the probability of observing the entire sequence following a particular state given that we’re in state $k$. (prob of end of seq given current state) is defined as:

$$b_k(i) = \sum_i e_i(x_{i+1}) a_k b_i(i + 1)$$

The only difference with the forward algorithm is that “$l$” now appears within the summation and therefore, the emission probabilities are different for every state, and cannot be extracted from the sum, giving the backward algorithm in Figure ??.

Putting them altogether, we now have $P(\pi_i = k|x)$ known as posterior decoding, which gives a new path that gives the most likely state at every position. The probability that $i^{th}$ state is $k$, given all emissions $x$.

$$P(k) = P(\pi_i = k|x) = \frac{f_k(i)b_k(i)}{P(x)}$$

Figure 8.3: Forward and Backward Algorithm

Figure 8.4: The Backward Algorithm
For classification, based on the entire sequence, if we’re trying to find out if a region is in fact a promoter or not, the Posterior decoding method works better because it’s more informative than the Viterbi path with summing over all path. Even though the Viterbi path is the most likely path it may share only a tiny portion of the probability. We might also want to make a prediction which in that case we use the posterior path. However, it may give us an invalid sequence of states! For example it might give two consecutive states that transition between them is not possible at all. So the actual probability of that path is zero because of an illegal transition.

All in all, if your application is to decode the whole genome you may prefer the Viterbi path, but if you want high accuracy at every position, you may prefer the posterior decoding.

8.3 Learning

We saw how to score and decode a sequence in two different ways.

In order to train an HMM, we have to learn the transition and emission probabilities. If the right answer is known for example if we’re given a genomic region of say a million nucleotides, and we have good experimental annotation of CpG islands, we are going to do Supervised Learning.

If instead we don’t know the right answer and we don’t know how frequent the CpG islands are, neither do we know the composition and where they are, and we just know that there exist some CpG islands, we wish to update our parameters of the model to maximize \( P(x|\theta) \) the sequence given the parameters \( \theta = \{\mathit{E}_i, \mathit{A}_{ij}\} \).

8.3.1 Supervised Learning

Estimating model parameters based on labeled data is actually trivial. Suppose that you are given a labeled sequence \( x_1, \ldots, x_N \), meaning the true hidden state sequence \( \pi_1, \ldots, \pi_N \) is known and want to estimate the parameters. We first define \( A_{kl} \) to be the number of times that a transition from state \( k \) to state \( l \) occurs in \( \pi \), and \( E_k(b) \) to be the number of times that state \( k \) in \( \pi \) emits \( b \) in \( x \). The parameters \( \theta \) that maximize \( P(x|\theta) \) is simply obtained by counting.

\[
a_{kl} = \frac{A_{kl}}{\sum_i A_{ki}} \quad (8.8a) \\
e_k(b) = \frac{E_k(b)}{\sum_c E_k(c)} \quad (8.8b)
\]

An example is shown in Figure ???. Now the problem is that in the above example, every time you have a T, you get a zero probability and this will bias interpretations which are due to over-fitting or small sample size. To deal with over-fitting, the solution is using pseudocounts. We can add pseudocounts representing our prior beliefs about the model parameters:

- \( A_{kl} = \# \) of times a \( k \rightarrow l \) transition occurs in \( \pi + r_{kl} \)
- \( E_k(b) = \# \) of times state \( k \) in \( \pi \) emits \( b \) in \( x + r_k(b) \)

Larger pseudocounts correspond to a strong prior belief. Consequently, small pseudocounts (\( \epsilon < 1 \)) are used to avoid 0 probabilities.

8.3.2 Unsupervised Learning

Unsupervised learning is estimating parameters based on unlabeled data, not knowing \( A_{kl} \) or \( E_k(b) \). If we have some guesses to the probabilities \( \theta \) for initialization, we can find the other parameters with parsing the data based on initial probabilities by Viterbi or Posterior decoding. Then we will have labeled data which can be used for supervised learning. Afterwards, the cycle repeats where we update our guess and so on. Now, where does the constraint come in into the learning? If we initialize differently do we end up with the same results? Yes because the data will bring down a false high probability that we might have guessed for a region. So the genome actually filters the data at each step. However, different initializations might lead to local maxima. There are two methods for learning:
Viterbi training: picking the best guess for the model and performing the Viterbi algorithm. It converges to a local maxima very rapidly.

Baum-Welch Learning: Learning over all paths by applying EM to HMMs.

The Viterbi training is a simple case. First pick the best-guess for model parameters and iterate the following steps until convergence.

1. Perform Viterbi to find $\pi^*$
2. Calculate $A_{kl}, E_k(b)$ according to $\pi^* +$ pseudocounts
3. Calculate the new parameters $a_{kl}, e_k(b)$

Expectation Maximization: The Baum-Welch Algorithm

E step: Estimate the expected probability of hidden labels given the current latest parameters and the observed unchanging sequence.

M step: Choose new ML sets of parameters over the distribution of sequence and labels given the current probabilistic assignments.

For example to count the transition matrix in Figure ??, we’re effectively using the forward and backward algorithm to find the entire probability capturing that position. What we end up with is known as the Baum-Welch algorithm, shown in Figure ??.

The time complexity is the number of iterations multiplied by the cost $O(K^2N)$. It guarantees to converge locally but not necessarily globally.

8.4 Current Research Directions

8.5 Further Reading

8.6 Tools and Techniques

8.7 What Have We Learned?

Bibliography
\[ A_{kl} = \sum_i P(\eta_i = k, \pi_i = l \mid x, \theta) = \sum_i \frac{f_i(i) a_{kl} \theta(x_{i-1}) b(l+1)}{P(x \mid \epsilon)} \]

Similarly,

\[ E_{kl}(b) = \frac{1}{P(x)} \sum_i \{i \mid x_i = b\} f_i(i) b_{kl}(i) \]

Figure 8.6: EM - New Parameters give probabilistic parse

---

**The Baum-Welch Algorithm**

*Initialization:*

Pick the best-guess for model parameters
(or arbitrary)

*Iteration:*

1. Forward
2. Backward
3. \( \Rightarrow \) Calculate new log-likelihood \( P(x \mid \theta) \) (E step)
4. Calculate \( A_{kl}, E_{kl}(b) \)
5. \( \Rightarrow \) Calculate new model parameters \( a_{kl}, \theta_{kl}(b) \) (M step)

**GUARANTEED TO BE HIGHER BY EXPECTATION-MAXIMIZATION**

Until \( P(x \mid \theta) \) does not change much

Figure 8.7: Baum-Welch Algorithm
CHAPTER
NINE

GENE IDENTIFICATION: GENE STRUCTURE, SEMI-MARKOV, CRFS

Jenny Cheng

List of figures

Intergenic DNA ??
Intron/exon splicing ??
Delineation of exons and open reading frames ??
Hidden Markov model for RNA splicing ??
Multiple lines of evidence for gene identification ??
HMMs with composite emissions ??
State diagram that considers direction of RNA translation ??
Conditional random fields: a discriminative approach conditioned on the input sequence ??
Examples of feature functions ??
Conditional probability score of an emitted sequence ??
A comparison of HMMs and CRFs ??

115
9.1 Introduction

Computational gene identification is the automatic annotation of genomic DNA to identify protein-coding regions. More than a tool for locating genes, computational gene identification seeks to provide an accurate model of the syntax of DNA.\(^1\)

9.2 Overview

This lecture talks about basic computational models to parse DNA into exon-intron structures, more specifically, to tag each of the ACTG nucleic acid base in a genomic DNA sequence as a protein-coding region or a non-protein-coding region.

9.3 Eukaryotic Gene Expression

In eukaryotes, not all of the genomic DNA sequences are transcribed. Regions of useful genetic information, such as genes, are typically clustered together, separated by intergenic regions, (region of the genome with few protein-coding genes).\(^2\)

![Intergenic DNA](Image)

Figure 9.1: Intergenic DNA

After transcription, before pre-mRNA can be made into proteins, the introns, regions within a gene that doesn't get translated into protein sequences, will have to be spliced out. Only the remaining transcribed exons, pieced together to form mRNA, will be translated into proteins. Here is an introduction to modeling this behavior of splicing out introns and keeping only the exons, based only on the specific genomic DNA sequences.

![Intron/Exon Splicing](Image)

Figure 9.2: Intron/Exon Splicing

9.4 Computational Gene Identification

The general assumptions for computational gene identification are exons are delineated by a sequence AG at the start of the exon and a sequence of GT at the end of the exon. For protein-coding genes, the start

---

codon (ATG) and the end codons (TAA, TGA, TAG) delineate the open reading frame.

9.5 Hidden Markov Model

A toy Hidden Markov Model, HMM, is a generative approach to model this behavior. Each emission of the HMM is one DNA base/letter. The hidden states of the model are intergenic, exon, intron. A better model would be to include hidden states DonorG and DonorT. The DonorG and DonorT states utilizes the information that exons are delineated by GT at the end of the sequence before the start of an intron.

The e in each state represents emission probabilities and the arrows indicate transition probabilities. Aside from the initial assumptions, additional evidence can help create a HMM to better model the behavior.

Combining all the lines of evidence can create a HMM with composite emissions.

A few assumptions are each new emission feature is independent of the rest. However, this creates the problem that with each new feature, the tuple increases in length, and the number of states of the HMM increases exponentially, leading to combinatorial explosion, which makes this scale poorly.

A better model takes into account the direction of translations, since the 5 strand and 3 strand are antiparallel. One idea by a student was to run the antiparallel separately in a HMM with fewer states, twice, instead of including additional states in the HMM. However, if the model yields overlapping regions, there’s no good way of resolving them.

9.6 Conditional Random Fields

Conditional Random Fields, CRF, are an alternative to HMM. Being a discriminative approach it doesn’t take into account the joint distribution of everything, like a HMM. The hidden states in a CRF are conditioned on the input sequence.\(^3\)

A feature function is like a score, returning a real-valued number as a function of its inputs. The conditional probability of the emitted sequence is its score divided by the total score of the hidden state. Examples of feature functions:

Each feature function is weighted, so that during the training, the weights can be set accordingly.

The feature functions can incorporate vast amounts of evidence without the Naive Bayes assumption of independence of all the evidence. However, training is much more difficult with CRFs than HMMs.

### 9.7 Other Methods

Semi-markov models generate a variable sequence length emissions, so it is not entirely memory-less transitions on the hidden states.

Max-min models are adaptations of support vector machines. These methods has not yet been applied to mammalian genomes. \(^4\)

---

\(^4\) For better understanding of SVM: [Link](http://dspace.mit.edu/bitstream/handle/1721.1/39663/6-034Fall-2002/OcwWeb/Electrical-Engineering-and-Computer-Science/6-034Artificial-IntelligenceFall2002/Tools/detail/svmachine.htm)
Figure 9.7: State diagram that considers direction of RNA translation

Figure 9.8: Conditional random fields: a discriminative approach conditioned on the input sequence

\[
\Pr(Y|X) = \frac{1}{Z(X)} \prod_{i} \psi(X, Y, i) \text{ where } Z(X) = \sum_{Y'} \prod_{i} \psi(X, Y', i)
\]

Figure 9.9: Examples of feature functions

9.8 Conclusion

Computational gene identification has a lot of practical significance as well as theoretical significance for the advancement of biological fields.

Two approaches:
6.047/6.878 Lecture 09: Gene Identification: Gene Structure, Semi-Markov, CRFs

\[
J(y_{i-1}, y_i, i, X) = \begin{cases} 
1 & \text{if } y_i = \text{exon and position } i \text{ is conserved in mouse} \\
0 & \text{otherwise} 
\end{cases}
\]

\[
f_2(y_{i-1}, y_i, i, X) = \begin{cases} 
1 & \text{if } y_i = \text{exon and position } i \text{ is conserved in rat} \\
0 & \text{otherwise} 
\end{cases}
\]

\[
f_3(y_{i-1}, y_i, i, X) = \# \text{ of mRNA sequences aligning to position } i \text{ (if } y_i = \text{exon; 0 otherwise)}
\]

Figure 9.10: Conditional probability score of an emitted sequence

### A comparison

<table>
<thead>
<tr>
<th>HMM</th>
<th>CRF</th>
</tr>
</thead>
</table>
| \[
\psi(X, Y, i) = a_{y_{i-1}, y_i} \cdot c_{y_i, x_i}
\] | \[
\psi(X, Y, i) = \exp \left( \sum_{k=1}^{n} \lambda_k f_k(y_{i-1}, y_i, i, X) \right)
\] |
| \[
\Pr(X, Y) = \prod_i \psi(X, Y, i)
\] | \[
\Pr(X, Y) = \exp \left( \sum_{k=1}^{n} \lambda_k f_k(y_{i-1}, y_i, i, X) \right)
\] |
| \[
\Pr(Y|X) = \frac{\Pr(X, Y)}{\Pr(X)} \quad \text{(Bayes’ law)}
\] | \[
\Pr(Y|X) = \frac{1}{\Pr(X)} \prod_i \psi(X, Y, i)
\] |
| \[
\Pr(X) = \sum_Y \prod_i \psi(X, Y, i)
\] | \[
\Pr(Y|X) = \frac{1}{Z(X)} \prod_i \psi(X, Y, i)
\] |

Q: How do we compute this efficiently?
A: Forward algorithm. CRFs have a direct analog (Viterbi too)

\[
\lambda_1 = 1, \quad f_1(y_{i-1}, y_i, i, x) = \log(a_{y_{i-1}, y_i} \cdot c_{y_i, x_i}) \quad \Rightarrow \{\text{HMM}\} \subset \{\text{CRF}\}
\]

Figure 9.11: A comparison of HMMs and CRFs

#### 9.8.1 HMM
- generative model
- randomly generating observable data, usually with a hidden state
- specifies a joint probability distribution
- \[
P(x, y) = P(x|y)P(y)
\]
- sometimes hard to model dependencies correctly
- hidden states are the labels for each DNA base/letter
- composite emissions are a combination of the DNA base/letter being emitted with additional evidence

#### 9.8.2 CRF
- discriminative model
- modeling dependence of unobserved variable y on an observed variable x
- \[
P(y|x)
\]
- hard to train without supervision
- more effective for when the model doesn’t require joint distribution
In practice, the results gene specification using CONTRAST, a CRF implementation, is about 46.2% at best. This is because in biology, there are a lot of exceptions to the standard model, such as overlapping genes, nested genes, and alternative splicing. Having models include all of those exceptions sometimes yield worse predictions. This is a tricky tradeoff. However, technology is improving and within the next five years, there will be more experimental data to fuel the development of computational gene identification, which in turn will help generate a better understanding of the syntax of DNA.

9.9 Current Research Directions

9.10 Further Reading

9.11 Tools and Techniques

9.12 What Have We Learned?

Bibliography
CHAPTER TEN

RNA FOLDING

Guest Lecture by
Stefan Washietl (wash@mit.edu)
Scribed by Sam Sinaei (samsinaei@mit.edu) (2010)

List of figures

Hierarchy of RNA Structure Complexity ??
RNA Secondary Structure ??
RNA Structure as Circle with Arcs ??
RNA Structure in Machine-Readable Dot-Bracket notation ??
RNA Structure as Matrix Representation ??
RNA Structure as a Mountain Plot ??
Base Pair Match Scoring Scheme ??
Nussinov Algorithm Recursion Formula ??
Nussinov Algorithm Setup Matrix ??
Nussinov Helper Array Recursion Algorithm ??
RNA Base Pair Stacking ??
Internal Substructures in Folded RNA ??
Zucker Algorithm ??
10.1 Introduction

RNA is a very important molecule and studying RNA has a lot of challenges in both computational and non-computational biology. In this section we would have a general overview on the topics that will follow. The aim for this lecture is to study the methods that can explain, or predict RNAs secondary structure and to do this we first look at RNA from a biological perspective and explain the known biological roles of RNA. Then we study the different methods which exist to predict RNA structure. There are two main approaches to the RNA folding problem; the first one is to predict RNA structure based on thermodynamic stability of the molecule, and look for a thermodynamic optimum. The second approach is based on probabilistic models which try to find the states of the RNA molecule in a probabilistic optimum. Finally, we can use evolutionary data in order to increase the confidence of our predictions by these methods.

10.1.1 History of the Field

10.1.2 Open Problems

10.2 Biological Overview

RNA biochemistry is consisted of a 5-carbon sugar, namely Ribose, which is attached to an Organic Base (Adenine, Uracil, Cytosine or Guanine). The biochemical difference of DNA and RNA comes in two places: First is the 5-carbon sugar which has no hydroxyl group in the 5 position, and second is the uracil presence in the RNA which is the nonmethylated form of thymine. The presence of the Ribose in RNA, makes its structure more flexible than DNA, therefore the RNA molecule is able to fold and make bonds within itself which makes the single stranded RNA much more stable than the single stranded DNA. The initial belief about the RNA was that it acts as an intermediate between the DNA code and the protein, which is true. However, in early 80s the discovery of catalytic RNAs (Ribozymes) expanded the perspective on what this molecule can actually do in living systems. Sidney Altman and Thomas Cech discovered the first ribozyme, RNase P which is able to cleave off the head of tRNA, which earned them the Nobel Prize for Chemistry in 1989. Self-splicing introns, namely group I introns, were also one of the foremost ribozymes that were discovered, and dont need any protein as catalysts to splice.

Single or double stranded RNA also serves as the information storage and replication agent in some Viruses. The RNA world Hypothesis [Gilbert 86] TODO: insert Bibtex citation @scribe: , which tries to explain the origin of life, relies on the fact that RNA can have both information storage, and catalytic activity at the same time, which are the fundamental characteristics that a living system needs to have. In short it says in the beginning RNA molecules were the first replicators, and since they where catalytic at the same time, it was possible for them to replicate without dependency on other molecules. Although to this day, there are no natural self-replicating RNA found in vivo, self-replicating RNA molecules have been created in lab by artificial selection. For example A chimeric construct of a natural ligase ribozyme with an in vitro selected template binding domain has been shown to be able to replicate at least one turn of an RNA helix.

Over evolution RNA has passed its information storage role to DNA, since it is more stable and less prone to mutation and acted as an intermediate between DNA and proteins, which took over some of RNAs catalytic role in the cell. This is why people sometimes refer to RNA as molecular fossils. Nevertheless, RNA still plays an important catalytic role in the living organisms, first and foremost the catalytic portion of the ribosome is consisted of RNA, meaning it is the main functional part of the ribosome complex. RNA also has regulatory roles in the cell, and basically serves as an agent for the cell to sense and react to the environment. Regulatory RNAs have different families, and one of the most important ones are Riboswitches which are involved in different levels of gene regulation. In some Bacteria some very important regulations are done by simple RNA regulation, one example is the Thermosensor in Listeria, a Riboswitch blocks the Ribosome at low temperature,( since the hydrogen bonds are more stable) the RNA forms a semi-double stranded conformation which doesn’t bind to the ribosome and turns the Ribosome off. At higher temperatures (37 C), the double strand opens up and allows Ribosome to attach to a certain region in the Riboswitch and hence the translation is possible.

Another famous Riboswitch is the adenine Riboswitch (and in general purine riboswitches) , which
regulate protein synthesis. For example the ydhl mRNA which has a terminator stem at the end and blocks it from translation, but when the Adenine concentration increases in the cell, it binds to the mRNA and changes its conformation such that the terminator stem disappears. There are other sorts of RNAs such as miRNA, a more modern variant of RNA so to speak, which their discovery unveiled a novel non-protein layer of gene regulation (e.g. the EVF-2 and HOTAIR miRNAs). EVF-2 is interesting because its transcribed from an ultra conserved enhancer, and separates from the transcription string by forming a hairpin, and thereafter returns to the very same enhancer (along with a protein Dlx-2) and regulates its activity. HOTAIR RNA induces changes in chromatin state, and regulates the methylation of Histones, which in turn silences the HOX-D cluster.

10.3 RNA Structure

So far we learned about different functions of RNA, and it should be clear by now how fundamental the role of RNA in living systems is. Since it is impossible to understand how RNA actually does all these activities in the cell, without knowing what it structure is, in this part we will look into RNAs structure.

RNA structure can be studied in three different levels:

1. **Primary** structure the sequence in which the bases in RNA are aligned. (A,U,G,C)
2. **Secondary** structure: the 2-D analysis of the [hydrogen] bonds between different parts of RNA. In other words, where it becomes double-stranded, where it forms a hairpin or loop and such forms.
3. **Tertiary** structure: is the complete 3-D structure of RNA, i.e. how the string bends, where it twists and such.

![Figure 10.1: Graphical representation of the hierarchy of RNA structure complexity](image)

As mentioned before, the ribose presence in the RNA makes it able to fold and create double-helixes with itself. There is not much to do to get the study primary structure, you have to sequence RNA and once you do that you basically know what the primary structure is. What we are mainly interested in is the secondary structure for RNA, where the loops and hydrogen bonds form and create the functional attributes of RNA. Ideally we would like to study the tertiary structure, because that is the final state of the RNA, and what gives it the real functionality, however, the tertiary structure is very hard to study computationally and is out of scope for this lecture.

Even though studying the secondary structure can be tricky, there are some simple ideas to that work quite well in predicting it. One might ask, if there is any point in studying the secondary structure if we cannot really predict the tertiary structure, and the answer is yes. Unlike proteins, in RNA most of the
[stabilizing] free energy for the molecule comes from its secondary structure (Rather than tertiary in case of proteins). RNAs initially fold into their secondary structure and then form their tertiary structure, and therefore there are very interesting fact that we can learn about a certain RNA molecule by just knowing its secondary structure. Finally, another great property of the secondary structure is that it is usually well conserved in evolution, which helps us to improve the secondary structure predictions and also to find ncRNA (non-coding RNA). There are 5 widely used representation for the RNA secondary structure:

Figure 10.2: The typical representation in textbooks. It clearly shows the secondary substructure in RNA.

Figure 10.3: As a graph drawing the back-bone as a circle and the base pairing as arcs within the circle. Note that the graph is outer-planar, meaning the arcs do not cross.

Formally: A secondary structure is a vertex labeled graph on $n$ vertices with an adjacency matrix $A = (a_{ij})$ fulfilling:

- $a_{i,i+1} = 1$ for $1 \leq i \leq n$ (continuous backbone)
- For each $i$, $1 \leq i \leq N$ there is at most one $a_{ij} = 1$ where $j \geq i + 1$ (a base only forms a pair with one other at the time)
- If $a_{ij} = a_{kl} = 1$ and $i < k < j$ then $i < l < j$ (ignore pseudo knots)

Figure 10.4: A machine readable dot-bracket notation, in which for each paired nucleotide you open a bracket (and close it when you reach its match) and for each unpaired element you have a dot.

### 10.4 RNA Folding Problem

Finally, we get to the point that we want to study the RNA structure. The goal here is to predict the secondary structure of the RNA, given its primary structure (or its sequence). The good news is we can find the optimal structure using dynamic programming. Now in order to set up our dynamic programming framework we would need a scoring scheme, which we would create using the contribution of each base pairing
to the physical stability of the molecule. In other words, we want to create a structure with minimum free energy, in our simple model we would assign each base pair an energy value.

Figure 10.7: Example of a scoring scheme for base pair matches. Note that G-U can form a wobble pair in RNA.

The optimum structure is going to be the one with a minimum free energy and by convention negative energy is stabilizing, and positive energy is non-stabilizing. Using this framework, we can use DP to calculate the optimal structure, first because this scoring scheme is additive, and second because we disallowed pseudo knots, which means we can divide the RNA into two smaller ones which are independent, and solve the problem for these smaller RNAs. So we want to find a DP matrix $E_{ij}$, in which we calculate the minimum free energy for subsequence $i$ to $j$.

### 10.4.1 Nussinov’s Algorithm

The recursion formula for this problem is first described by Nussinov[1978], hence it is known as the Nussinov algorithm.

More informally it calculates the best substructure for the sub-sequences and then builds up to larger sequences until it finds the structure of the whole sequence. There are basically only two cases to get from one step to the next in the recursion. Either the newly added base is unpaired or it is paired with some base $k$. For the latter case the base pair $(i,k)$ divides the problem into two subproblems which can be then recursively solved the same way. Below is example for the DP Matrix after running this algorithm:

Of course when you calculate the minimum free energy, you are interested in the sequence which corresponds to this particular energy, a helper matrix is filled to backtracking over the sequence, here is the code for the backtracking:

![Figure 10.5: A matrix representation, in which you have a dot for each pair.](image)

![Figure 10.6: Mountain plot, in which for pairs you go one step up in the plot and if not you go one step to the right.](image)

![Figure 10.7: Example of a scoring scheme for base pair matches. Note that G-U can form a wobble pair in RNA.](image)
Figure 10.8: The recursion formula for Nussinov algorithm, along with a graphical depiction of how it works.

\[
E_i = \min \left\{ E_{i+1,j}, \min_{k \neq i} \{ E_{i+1,k-1} + E_{k+1,j} + \beta \} \right\}
\]

- \( E_i \) : Minimum energy of subsequence \( i...j \)
- \( \beta \) : Energy contribution of pair \( (i,j) \)
- \( \Pi_j \) : 1 if bases \( i \) and \( j \) can pair and 0 otherwise.

Figure 10.9: The algorithm starts by assigning the diagonal for the matrix as 0 (since you cannot pair with yourself) and then works through the recursion up and right, the minimum free energy is the top-rightmost element in the matrix. The minimum length for a loop is 1 here (usually 3). As \( i \) counts backward from \( n \) to 1, \( j \) counts from 1 to \( n \). In this example we simply assign -1 for a pair, and 0 for a non-pair.

Figure 10.10: The helper array \( K_{ij} \) is filled during the recursion that holds the optimal secondary structure when \( k \) is paired with \( i \) for a sub-sequence \( i...j \). If \( i \) is unpaired in the optimal structure, \( K_{ij} \) is filled during the recursion that holds the optimal secondary structure when \( k \) is paired with \( i \) for a sub-sequence \( i...j \). If \( i \) is unpaired in the optimal structure, \( K_{ij} = 0 \).

Since this model is very simplistic, there are some limitations to it. Most importantly, stacking interaction between neighboring is a very important factor (even more important than the hydrogen bonds) in RNA folding which is not considered by the Nussinov model.

Therefore people have thought of methods to integrate such biophysical factors into our prediction. One improvement for instance is that instead of assigning energies to single base pairs, we assign them to faces of the graph (structural elements in ? ?). In order to find out the total energy of the structure, we have to find the free energy of each substructure, and simply add them up. The stacking energies can be calculated by melting oligonucleotides experimentally.
Figure 10.11: Stacking between neighboring base pairs is RNA. The flat aromatic structure of the base causes quantum interactions between stacked bases and changes its physical stability.

Figure 10.12: Various internal substructures in a folded RNA. A hairpin is consisted of a terminal loop connected to a paired region, an internal loop is an unpaired region within the paired region. A Bulge is a special case of an interior loop with a single mis-pair. A Multi loop is a loop which consists of multiple of these components (in this example two hairpins and a paired region, all connected to a loop).

10.4.2 Zucker Algorithm

The Nussinovs variant, which includes stacking energies to calculate the RNA structure is called Zuker algorithm. Like Nussinovs, it assumes that the optimal structure is the one with the lowest equilibrium free energy. Nevertheless, it is including the total energy contributions from the various substructures which is partially determined by stacking energy. Some modern RNA folding algorithms use this algorithm for RNA structure predictions. In Zuker algorithm, we have four different cases to deal with. Shows a graphical outline of the decomposition steps. The procedure requires four matrices. \( F_{ij} \) contains the free energy of the overall optimal structure of the subsequence \( x_{ij} \). The newly added base can be unpaired or it can form a pair. For the latter case, we introduce the helper matrix \( C_{ij} \), that contains the free energy of the optimal substructure of \( x_{ij} \) under the constraint that \( i \) and \( j \) are paired. This structure closed by a base-pair can either be a hairpin, an interior loop or a multi-loop.

The hairpin case is trivial because no further decomposition is necessary. The interior loop case is also simple because it reduces again to the same decomposition step. The multi-loop step is more complicated. The energy of a multi loop depends on the number of components, i.e. substructures that emanate from the loop. To implicitly keep track of this number there is need for additional two helper matrices. \( M_{ij} \) holds the free energy of the optimal structure of \( x_{ij} \) under the constraint that \( x_{ij} \) is part of a multi loop with at least one component. \( M^1_{ij} \) holds the free energy of the optimal structure of \( x_{ij} \) under the constraint that \( x_{ij} \) is part of a multi-loop and has exactly one component closed by pair \((i,k)\) with \( i < k < j \). The idea is to decompose a multi loop in two arbitrary parts of which the first is a multi-loop with at least one component and the second a multi-loop with exactly one component and starting with a base-pair. These two parts corresponding to \( M \) and \( M^1 \) can further be decomposed into substructures that we already know, i.e. unpaired intervals, substructures closed by a base-pair, or (shorter) multi-loops. (The recursions are also summarized in ??.

In reality, in room temperature (or cell temperature) RNA is not actually in one single state, but rather it varies in a Thermodynamic ensemble of structure. Base pairs can break their bonds quite easily, and although we might find an absolute optimum in terms of free energy, it might be the case that there is another sub-optimal structure which is very different from what we predicted and has an important role in the cell. To fix the problem we can calculate the base pair probabilities to get the ensemble of structures,
Figure 10.13: $F$ describes the unpaired case, $C$ is described by one of the three conditions: hairpin, interior loop, or a composition of structures i.e. a multi loop. $M^1$ is a multi loop with only one component, where are $M$ might have multiple of them. The $|$ icon is notation for or.

and then we would have a much better idea of what the RNA structure probably looks like. In order to do this, we utilize the Boltzman factor:

$$\text{Prob}(s) = \frac{\exp(-\Delta G(s)/RT)}{Z}$$

Which gives us the probability of a given structure, in a thermodynamic system. We need to normalize the temperature using the partition function $Z$, which is the weighted sum of all structures, based on their Boltzman factor:

$$Z = \sum_s \exp(-\Delta G(s)/RT)$$

We can also represent this ensemble graphically, using a dot plot to visualize the base pair probabilities. To calculate the specific probability for a base pair $(i,j)$, we need to calculate the partition function, which is given by the following formula:

$$\rho_{ij} = \frac{\bar{Z}_{i,j} \exp(-\beta_{ij}/RT)}{Z}$$

To calculate $Z$ (the partition function over the whole structure), we use the recursion similar to the Nussinovs Algorithm (known as McCaskill Algorithm). The inner partition function is calculated using the formula:

$$Z_q = Z_{i+1,j} + \sum_{k=i+1}^{j-1} Z_{i+1,k-1} Z_{k+1,j} \exp(-\beta_{ik}/RT)$$

With each of the additions corresponding to a different split in our sequence as the next figure illustrates. Note that the addition are multiplied to the energy functions since it is expressed as an exponential. Similarly the outer partition function is calculated with a the same idea using the formula: corresponding to different splits in the area outside the base pairs $(i,j)$. 

130
10.5 RNA Evolution

It is quite interesting to know about the evolution of RNA structure, since it unveils valuable data, and can also give us hints to refine our structure predictions. When we look into functionally important RNAs over time, although their nucleotides have changed at some parts, but the structure is well-conserved. In RNA there are a lot of compensatory or consistent mutations. In a consistent mutation the structure doesn’t change e.g. an AU pair mutates to form a G pair. In a compensatory mutation there are actually two mutations, one disrupts the structure, but the second mutation restores it, for example an AU pair’s changes to a CU which do not pair well, but in turn the U mutates to a G to restore a CG pair. In ideal world, if we have this knowledge, this is the key to predict the RNA structure, since evolution never lies. To do so we can calculate the mutual information content for two different RNAs and compare them. In other words, you compare the probability of two base pair structures agreeing randomly vs. if they have evolved to conserve the structure. The Mutual information content is calculated via this formula:

\[
M_{ij} = \sum_{X,Y} f_i(X) \log \frac{f_i(X)}{f_j(X)f_j(Y)}
\]

If we normalize these probabilities, and store the MI in bits, we can plot it in a 3D model and track the evolutionary signatures. In fact, this was the method for determining the structure of ribosomal RNAs long before they were found by crystallography, and it turned out to be very accurate for this case. The problem in reality is that we don’t have so much information, so what we usually do is we combine the folding prediction methods with phylogenetic information in order to get a reliable prediction. The most common way to do this is to combine to Zuker algorithm with some covariance scores. So for example we add stabilizing energy if we have a compensatory mutation, and destabilizing energy if we have a single nucleotide mutation somewhere.

10.6 Probabilistic Approach to the RNA Folding Problem

RNA-coding sequence inside the genome Finding RNA-coding sequences inside the genome is a very hard problem. However there are ways to do it. One way is to combine the thermodynamic stability information, with a normalized RNAfold score and then we can do a Support Vector Machine (SVM) classification, and compare the thermodynamic stability of the sequence to some random sequences of the same GC content and the same length, and see how many standard deviations is the given structure more stable that the expected value. Then we can combine it with the evolutionary measure and see if the RNA is more conserved or not. This gives us (with relative accuracy) and idea if the genomic sequence is actually coding an RNA.
So far we have studied only half of the story, although the thermodynamic approach is a good way (and the classic way) of folding the RNAs, some part of the community likes to study it from an absolutely different aspect. Let’s assume for know that we don’t know anything about physics of RNA or the Boltzman factor. Instead, look into the RNA as a string of letters for which we want to find the most probable structure.

We have already learned about the Hidden Markov Models in the previous lectures, which is a nice way to make predictions about the hidden states of a probabilistic system. The question is can we use Hidden Markov models for the RNA folding problem? And the answer is yes. We can represent the RNA structure as a set of hidden states of dots and brackets (recall the dot-bracket representation of RNA in part 3). There is an important observation to make here, the positions and the pairings inside the RNA are not independent, so we cannot simply have a state of an opening bracket without any considerations of the events that are happening downstream. Therefore we need to extend the HMM framework to allow for nested correlations. Fortunately, the probabilistic framework to deal with such problem already exists, known as Stochastic context-free grammar (SCFG).

### Context Free Grammar in a nutshell

You have:
- Finite set of non-terminal symbols (states) e.g. \{A, B, C\} and terminal symbols e.g. \{a, b, c\}
- Finite set of Production rules. e.g. \{A → aB, B → AC, B → aa, → ab\}

You want: Find a way to get from one state to another (or to a terminal). \(A → aB → aAC → aaaaC → aaaaab\) In a stochastic CFG, each relation has a certain probability. e.g. \(P(B → AC) = 0.25\) \(P(B → aa) = 0.75\)

Phylogenetic evaluation is easily combined with SCFGs, since there are many probabilistic models for phylogenetic data. The Probabilistic models are not discussed in detail in this lecture but the following picture basically gives an analogy between the Stochastic models and the methods that we have see so far in the class.

- Analogy to thermodynamic folding:
  - CYK ↔ Minimum Free energy (Nussinov/Zuker)
  - Inside/outside algorithm ↔ Partition functions (McCaskill)

- Analogy to Hidden Markov models:
  - CYK Minimum ↔ Viterbi’s algorithm
  - Inside/outside algorithm ↔ Forward/backwards algorithm

- Given a parameterized SCFG \((\Theta, \Omega)\) and a sequence \(x\), the Cocke-Younger-Kasami (CYK) dynamic programming algorithm finds an optimal (maximum probability) parse tree \(\hat{\pi}\):
  \[\hat{\pi} = \arg\max \text{Prob}(\pi, x|\Theta, \Omega)\]

- The Inside algorithm, is used to obtain the total probability of the sequence given the model summed over all parse trees,
  \[\text{Prob}(x|\Theta, \Omega) = \Sigma \text{Prob}(x, \pi|\Theta, \Omega)\]

### 10.7 High Throughput methods to detect non-coding RNAs

There are plenty of RNAs inside the cell aside from mRNAs, tRNAs and rRNAs. The question is what is the relevance of all this non-coding RNA? Some believe it is noise resulting from experiment, some think its just biological noise that doesn’t have a meaning in the living organism. On the other hand some believe junk RNA might actually have an important role as signals inside the cell and all of it is actually functional, the truth probably lies somewhere in between. ?? summarizes the methods to detect transcribed RNA.
### Table 10.1: Three important methods to find transcribed RNA

<table>
<thead>
<tr>
<th>Technology</th>
<th>Tiling Microarray</th>
<th>cDNA or EST sequencing</th>
<th>RNA-Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology specifications</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Principle</td>
<td>Hybridization</td>
<td>Sanger sequencing</td>
<td>High-throughput sequencing</td>
</tr>
<tr>
<td>Resolution</td>
<td>From several to 100 bp</td>
<td>Single base</td>
<td>Single base</td>
</tr>
<tr>
<td>Throughput</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Reliance on genomic sequence</td>
<td>Yes</td>
<td>No</td>
<td>In some cases</td>
</tr>
<tr>
<td>Background noise</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simultaneously map transcribed regions and gene expression</td>
<td>Yes</td>
<td>Limited for gene expression</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamic range to quantify gene expression level</td>
<td>Up to a few hundredfold</td>
<td>Not practical</td>
<td>&gt; 5000 fold</td>
</tr>
<tr>
<td>Ability to distinguish different isoforms</td>
<td>Limited</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Ability to distinguish allele expression</td>
<td>Limited</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Practical issues</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required amounts of RNA</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Cost for mapping transcriptomes of large genomes</td>
<td>High</td>
<td>High</td>
<td>Relatively low</td>
</tr>
</tbody>
</table>

**Current research:** There are conserved regions in the genome that do not code any proteins, and now Stefans et al. are looking into them to see if they have structures that are stable enough to form functional RNAs. It turns out that around 6% of these regions have hallmarks of good RNA structure, which is still 30000 structural elements. The group has annotated some of these elements, but there is still a long way to go. A lot of miRNAs, snoRNAs have been found and of course lots of false positives. But there exciting results coming up in this topic! so the final note is, it’s a very good area to work in!

### 10.8 Current Research Directions

### 10.9 Further Reading

### 10.10 Tools and Techniques

### 10.11 What Have We Learned?

**Bibliography**


CHAPTER

ELEVEN

LARGE INTERGENIC NON-CODING RNAs

Guest lecture by John Rinn
Scribed by Eli Stickgold (2010)

List of Figures

TODO: missing @scribe: insert images

135
11.1 Introduction

11.1.1 The Big Question? Why are We Doing This?

The same genome is present in every cell; particular state the genome takes on creates all the different cell
types. (“Spatial patterns of cellular differentiation”). Cell types can be determined by histone modification
or DNA Methylation (a binary code, with the two states being euchromatic and heterochromatic).

Can now scan cells for histone modifications as if they were barcodes to detect cell types. RNAs can
influence setup of cell.

DNA methylation can be thought of as “Bar codes” for identification.

FAQ

Q: How did bar codes get there in the first place? How did the differentiation happen?

A: Non-coding RNAs are heavily involved in this process. These RNA strands are called Large
Intergenic Non-Coding RNAs (lincRNAs).

A quick history of RNA:

• 1975: A lab testing relative levels of RNA and DNA in bull sperm discovers twice as much RNA as
DNA.

• 1987: After automated sequencing developed, weird non-coding RNAs are first found.

• 1988: RNA is proved to be important for maintaining chromosome structures, via chromatin archi-
tecture.

• 1990s: A large number of experiments start to research RNA 2000: Study shows Histone-methyl
transferases depend on RNA, as RNAase causes the proteins to delocalize.

Transcription is a good proxy of what’s active in the cell and what will turn into protein. Microarrays
led to the discovery of twice as many non-coding genes as coding genes initially; now we know the ratio is
even far higher than this.

Ed Lewis says that some early insights into the existence of protein-coding genes as genetic elements.

11.2 Noncoding RNAs from Plants to Mammals

Basic Cycle: large RNA gets chopped up into small RNAs (siRNAs) RNA use by category:

Protists: RNA is used as a template to splice out DNA (RNA-dependent DNA elimination and splicing)

mRNA and DNA in nucleus: DNA chopped and recombined based on gaps in mRNA (”quirky phenom-
ena”)

Plants: RNA-dependent RNA polymerase, where the polymerase takes template of RNA and make a copy
of it, is available in plants but not humans, and can make small RNAs. Mammals have at most one
copies. Very different than RNA polymerase and DNA polymerase in structure. From this, we know
that plants do DNA methylation with noncoding RNA.

Flies: use RNAs for an RNA switch; coordinated regulation of hox gene requires noncoding RNA.

Mammals: Non-coding RNAs can form triple helices, guide proteins to them; chromatin-modifying com-
plexes; involved in germ line; guide behaviour of transcription factors.
In general, DNA-binding domain on proteins can somewhat easily bind RNA since RNA/DNA have similar structure.

For the rest of this talk, we focus on specifically lincRNA, which we will define as RNA larger than 200 nucleotides.

Types of lincRNA include:

**Antisense:** Very important, these overlap protein-coding genes

**Intronic:** Solely within introns

**Bidirectional:** Very common, these RNAs use same promoters going off in different directions

**Intergenic:** Discussed in detail in next section

### 11.3 Integerneic Non-coding RNAs: missing lincs in Stem/Cancer cells?

One RNA can “crumple entire chromosome” X chromosome randomly inactivated by ncRNAs: colour gene is on X, causes colour patterns on cat. All organisms use hox genes to lay out body pattern; skin cells are different in different parts of the body; hox TF codes can be used to tell which skin cell came from where. ncRNAs cause this.

RNA is important for getting polychrome complex to chromosome ncRNAs can activate downstream genes in Cis, opposite in trans; Xist does the same thing.

### 11.4 Technologies: in the wet lab, how can we find these?

How would we find ncRNAs? We have about 20-30 examples of ncRNAs with evidence of importance, but more are out there. Chromatin state maps (from ENCODE, chip-seq) can be used to find transcriptional units that do not overlap proteins. We can walk along map and look for genes (look by eye at chromatin map to find ncRNAs). Nearly 90% of time such a signature is found, RNA will be transcribed from it. We can validate this through northern blot.

When looking at a chromatin map to find ncRNAs, we are essentially looking through the map with a window of a given size and seeing how much signal vs. noise we are getting, compared to what we might expect from a random-chance hypothesis. As both large and small windows have benefits, both should be used on each map section. Larger windows encapsulate more information; smaller windows are more sensitive.

After finding integenic regions, we find conserved regions.

We check if new regions are under selective pressure; fewer mutations in conserved regions. If a nucleotide never has a mutation between species, it’s highly conserved.

linc-RNAs are more conserved than introns, but less conserved than protein-coding introns, possibly due to non-conserved sequences in loop regions of lincRNAs.

Finding what lincRNAs’ functions are: “Guilt by association”: We can find proteins that correlate with particular lincRNA in terms of expression; lincRNAs are probably correlated to a particular pathway. In this way, we acquire a multidimensional barcode for each lincRNA (what it is and is not related to). We Can cluster lincRNA signatures and identify common patterns. Lots have to do with cell cycle genes. (This approach works 60-70% of the time)

#### 11.4.1 Example: p53

Independent validation: we use animal models, where one is a wild-type p53, and one is a knockout. We induce p53, then ask if lincRNAs turn on. 32 of 39 lincRNAs found associated with p53 were temporally induced upon turning on p53.

One RNA in particular sat next to a protein-coding gene in the p53 pathway. We tried to figure out if p53 bound to promoter and turned it on. To do this, we cloned the promoter of lincRNA, and asked does p53 turn it on? We IPed the p53 protein, to see if it associated with the lincRNA of the promoter. It turned...
out that lincRNA is directly related to p53 - p53 turns it on. P53 also turns genes off - certain lincRNAs act as a repressor.

From this example (and others), we start to see that RNAs usually have a protein partner. RNA can bring myriad of different proteins together, allowing the cell lots of diversity. In this way its similar to phosphorylation. RNAs bind to important chromatin complexes, and is required for reprogramming skin cells into stem cells.

11.5 Practical topic: RNAseq

Even with RNAseq, estimating transcript abundance and doing de novo assembly is extremely hard.

Transcript abundance: Measured in fragments per kilobase - number of reads of transcript per million reads in read run. This is a rough, quick way of finding most differentially expressed RNA genes.

Isoform abundance is a problem - which isoform did each transcript come from? CUFFLINCS gives a log-likelihood surface with a unique global maximum based on likelihood for a given isoform. Using sampling, we can climb up to find the maximum, using gradient/covariance we can assign confidence to each region.

Experimental determination of RNA structure: enzymes and chemicals chop up structured regions or unstructured regions. If we titrate the enzymes to make one cut per molecule, we can have sequence reads seeing where cuts occurred.

11.6 Current Research Directions

11.7 Further Reading

11.8 Tools and Techniques

11.9 What Have We Learned?

Bibliography

CHAPTER

TWELVE

SMALL RNA

TODO: missing @scribe: who was this scribed by?

List of Figures

TODO: missing @scribe: insert images
12.1 Introduction

12.2 Current Research Directions

12.3 Further Reading

12.4 Tools and Techniques

12.5 What Have We Learned?

Bibliography
Part III

Gene and Genome Regulation
CHAPTER

THIRTEEN

GENE REGULATION 1 – GENE EXPRESSION CLUSTERING

Tahin Syed (2010)
Barrett Steinberg and Brianna Petrone (2009)
Mia Y. Qia (2008)

List of figures

Matrix Gene Expression Values ??
Iterations of K-Means Clustering ??
K Means Clustering Final Results ??
13.1 Introduction

After finishing gene finding methods and RNA folding in the past week, we are moving into the third module of the course focusing on gene regulation and networks. Here, we consider the problem of discovering previously unknown similarities or patterns within high dimensional datasets. Finding structure in such data sets allows us to derive conclusions about the process underlying the observations. In this first lecture in the module, we approach this problem through the application of clustering techniques.

One important distinction to be made early on is the difference between clustering data and classifying. Classification is a supervised learning method in which we already have labeled objects and wish to derive rules to assign labels to new objects. Clustering, a form of unsupervised learning, is the process of grouping unlabeled points into clusters by using some distance measure to determine the similarity or nearness of data points in some space.

13.1.1 History of the Field

13.1.2 Open Problems

13.2 Motivation

We can find many applications of clustering that are specific to computational biology. Take for instance the ENCODE/modENCODE project, which looks at every single aspect of activity associated with the genome across different stages of development and various cell lines. The data gathered by this effort includes properties of nucleosomes, histone modifications, dynamics of replication, processing of RNA, binding of transcription factors, and more. Using all of this data, it is possible to make functional predictions about genes through the use of clustering.

For example, consider expression profiles of many genes taken at various developmental stages. Clustering may show that certain sets of genes line up (i.e. show the same expression levels) at various stages. This may indicate that this set of genes has common expression or regulation and we can use this to infer similar function. Furthermore, if we find a uncharacterized gene in such a set of genes, we can reason that the uncharacterized gene also has a similar function through guilt by association. Chromatin marks and regulatory motifs can be used to predict logical relationships between regulators and target genes in a similar manner. This sort of analysis can lead to models that allow us to predict gene expression. These models could be applied to changing the regulation of a particular gene, predicting how a disease state arose, or aiding in targeting genes to a part of the body, if we know what kind of regulation goes on there.

13.3 Microarrays

Microarray analysis is one of the most common techniques to which clustering is applied. The most intuitive way to investigate a certain phenotype is to measure the functional proteins present at a given time in the cell. However, given the difficulty of measuring protein, due to their varying locations, modifications, and contexts in which they are found, as well as due to the incompleteness of the proteome, mRNA expression levels are commonly used as a good approximation, as it is much easier to measure and allows 1000s of genes to be measured in one experiment. Furthermore, given the Central Dogma of Molecular Biology, it is more desirable to measure mRNA since we are measuring the regulation that occurs at the level of the genome. By measuring protein, we would be combining two regulatory steps.

13.3.1 Technology/Biological Methods

The basic principle behind microarrays is the hybridization of complementary DNA fragments. To begin, short segments of DNA, known as probes, are attached to a solid surface, commonly known as a gene chip. Then, the RNA population of interest, which has been taken from a cell is reverse transcribed to cDNA (complementary DNA) via reverse transcriptase, which synthesizes DNA from RNA using the poly-A tail as a primer. For intergenic sequences which have no poly-A tail, a standard primer can be ligated to the ends of the mRNA. The resulting DNA has more complementarity to the DNA on the slide than the RNA. The
cDNA is then washed over the chip and the resulting hybridization triggers the probes to fluoresce. This can be detected to determine the relative abundance of the mRNA in the target.

Two basic types of microarrays are currently used. Affymetrix gene chips have one spot for every gene and have longer probes on the order of 100s of nucleotides. On the other hand, spotted oligonucleotide arrays tile genes and have shorter probes around the 10s of bases.

There are numerous sources of error in the current methods and future methods seek to remove steps in the process. For instance, reverse transcriptase may introduce mismatches, which weaken interaction with the correct probe or cause cross hybridization, or binding to multiple probes. One solution to this has been to use multiple probes per gene, as cross hybridization will be different for each gene. Still, reverse transcription is necessary due to the secondary structure of RNA. The structural stability of DNA makes it less probable to bend and not hybridize to the probe. The next generation of technologies, such as RNA-Seq, sequences the RNA as it comes out of the cell, essentially probing every base of the genome.

13.3.2 Gene Expression Matrix

The amount of data generated by a microarray is enormous. For example, taking the output of a gene prediction model and making probes for every single one, detects the expression of every gene in the cell. Furthermore, performing experiments across conditions, time courses, stages of development, phenotype, or any other factor increases the amount of data. The result is often represented as an \( m \times n \) expression matrix, profiling the expression levels of \( m \) genes across \( n \) experiments. Clustering gene expression data enables data visualization to identify what different cancers look like, find functional similarities between genes, and can facilitate the development of a gene regulatory network model.

These matrices can be clustered hierarchically showing the relation between pairs of genes, pairs of pairs, and so on, creating a dendrogram in which the rows and columns can be ordered using optimal leaf ordering algorithms.

This predictive and analytical power is increased due to the ability of biclustering the data; that is, clustering along both dimensions of the matrix. The matrix allows for the comparison of expression profiles of genes, as well as comparing the similarity of different conditions (i.e. diseases). A challenge, though, is the curse of dimensionality. As the space of the data gets larger and large, the clustering of the points diminishes. Sometimes, the data can be reduced to lower dimensional spaces to find structure in the data using clustering to infer which points belong together based on proximity.

![Figure 13.1: A sample matrix of gene expression values, represented as a heatmap and with hierarchal clusters.](image)

Interpreting the data can also be a challenge, since there may be other biological phenomena in play. For example, protein coding exons have higher intensity, due to the fact that introns are rapidly degraded. At
the same time, not all introns are junk and there may be ambiguities in alternative splicing. There are also cellular mechanisms that degrade aberrant transcripts through non-sense mediated decay.

### 13.4 Clustering Algorithms

There are two types of clustering algorithms, partitioning and agglomerative. Partitional clustering divides objects into non-overlapping clusters so that each data object is in exactly one subset. Alternatively, agglomerative clustering methods yield a set of nested clusters organized as a hierarchy representing structures from broader to finer levels of detail.

#### 13.4.1 K-Means Clustering

The $k$-means algorithm clusters $n$ objects based on their attributes into $k$ partitions. **TODO: clarify**

The $k$-means algorithm:

1. Assumes a fixed number of clusters, $k$

2. **Initialization:** Randomly initialize the $k$ means $\mu_k$ associated with the clusters and assign each data point $x_i$ to the nearest cluster, where the distance between $x_i$ and $\mu_k$ is given by $d_{i,k} = (x_i - \mu_k)^2$.

3. **Iteration:** Recalculate the centroid of the cluster given the points assigned to it

$$
\mu_k(n + 1) = \sum_{x_i \in k} \frac{x_i}{|x_k|}
$$

where $x_k$ is the number of points with label $k$. Reassign data points to the $k$ new centroids by the given distance metric. The new centers are effectively calculated to be the average of the points assigned to each cluster.
4. **Termination**: Iterate until convergence or until a user-specified number of iterations has been reached (the iteration may be trapped at some local maxima or minima).

There are several methods for choosing $k$: simply looking at the data to identify potential clusters or iteratively trying values for $n$, while penalizing model complexity. You can always make better clusters by increasing $k$, but at some point you begin overfitting the data.

We can also think of k-means as trying to minimize a cost criterion associated with the size of each cluster, where the cost increases as the clusters get less compact.

### 13.4.2 Fuzzy K-Means Clustering

In fuzzy clustering, each point has a probability of belonging to each cluster, rather than completely belonging to just one cluster. Fuzzy k-means specifically tries to deal with the problem where points are somewhat in between centers or otherwise ambiguous by replacing distance with probability (which of course could be some function of distance, such as having probability relative to the inverse of the distance), and by using a weighted centroid based on those probabilities. Similar processes of initialization, iteration, and termination are used as in k-means. The resulting clusters are best seen as probabilistic distributions rather than a hard assignment of labels. One should realize that k-means is a special case of fuzzy k-means when the probability function used is simply 1 if the data point is closest to a centroid and 0 otherwise.

The fuzzy k-means algorithm:

1. Assume a fixed number of clusters, $k$

2. **Initialization**: Randomly initialize the $k$ means $\mu_k$ associated with the clusters and compute the probability that each data point $x_i$ is a member of a given cluster $k$, $P(point | x_i, k)$.

3. **Iteration**: Recalculate the centroid of the cluster as the weighted centroid given the probabilities of membership of all data points $x_i$,

$$
\mu_k(n+1) = \frac{\sum_{x_i \in k} x_i \cdot P(\mu_k | x_i)^b}{\sum_{x_i \in k} P(\mu_k | x_i)^b}
$$

4. **Termination**: Iterate until convergence or until a user-specified number of iterations has been reached (the iteration may be trapped at some local maxima or minima)
13.4.3 K-Means as a Generative Model

A generative model is a model for randomly generating observed data, given some hidden parameters. While a generative model is a probability model of all variables, a discriminative model provides a conditional model only of the target variable(s) using the observed variables.

When we perform clustering, we are assuming something about the underlying data. In the case of k-means and fuzzy k-means, we are assuming that a set $k$ centers (parameters) generate the data points using a Gaussian distribution for each $k$, potentially generating a stochastic representation of the data. In the case where $k = 2$, this can be thought of as flipping a coin to choose one of the two centers, then randomly placing a point, according to a Gaussian distribution, somewhere near the center. Unfortunately, since k-means assumes independence between the axes, covariance and variance are not accounted for using $k$-means, so models such as oblong distributions are not possible. In the clustering processes discussed containing a set of labeled data points $x$, we want to choose the most probable center (parameter) for $x$; that is, we want to maximize the probability of the clustering model. This is the maximum likelihood setting of $\mu_k$, given the data. Given a set of $x_i$ and all labels $k$, we can find the maximum likelihood $\mu_k$ as follows:

\[
\arg\max_{\mu} \left\{ \log \prod_i P(x_i \mid \mu) \right\} = \arg\max_{\mu} \sum_i \left\{ -\frac{1}{2} (x_i - \mu)^2 + \log \left( \frac{1}{\sqrt{2\pi}} \right) \right\} \\
= \arg\min_{\mu} \sum_i (x_i - u)^2
\]

Probability in this case is a function of distance of each data point from each center. After that, we want to find the best new parameter, the maximum likelihood for the next iteration of the algorithm using the same processes.

These same principles apply in reverse to determine labels from known centers, similarly using the argmin function. In this case, we attempt to find the best label that maximizes the likelihood of the data. We find that this is the same as simply finding the nearest center:

If neither labels nor centers are known, a common solution to estimate both is to start with $k$ arbitrary centers, calculate the most likely labels given the centers, use these labels to choose new centers, and iterate until a local maximum of probability is reached.

K-means can be seen as an example of EM (expectation maximization algorithms), where expectation consists of estimation of hidden labels, $Q$, and maximizing of expected likelihood occurs given data and $Q$. Assigning each point the label of the nearest center corresponds to the E step of estimating the most likely label given the previous parameter. Then, using the data produced in the E step as observation, moving the centroid to the average of the labels assigned to that center corresponds to the $M$ step of maximizing the likelihood of the center given the labels. This case is analogous to Viterbi learning. A similar comparison can be drawn for fuzzy $k$-means, which is analogous to Baum-Welch from HMMs.

EM is guaranteed to converge and guaranteed to find the best possible answer, at least from an algorithmic point of view. The notable problem with this solution is that of local maxima of probability distributions in which only uphill movement is possible. Thus an absolute maximum may never be determined. This problem may be hopefully avoided by attempting multiple initializations to better determine the landscape of probabilities.

(didn’t get to hierarchical clustering/hyper geometric distribution in this lecture is covered in next set of notes)
13.5 Current Research Directions

13.6 Further Reading

13.7 Tools and Techniques

13.8 What Have We Learned?

Bibliography

CHAPTER

FOURTEEN

GENE REGULATION 2 – CLASSIFICATION

Fulton Wang (fultonw@mit.edu)
Salil Desai
David Charlton
Kevin Modzelewski
Robert Toscano

List of figures
14.1 Introduction

In the previous lecture we looked at clustering, which is a way to capture unknown structure in a data set. This is an example of unsupervised learning, because the algorithm tries to find an effective classification based on very few prior assumptions. This lecture looks at the other side of the coin, supervised learning: if we have some preexistent hints about the inputs structure, we want to convert that into a rule that lets us classify new data sets.

There are two ways to do classification. The two ways are analogous to the two ways we performed gene finding: HMM, which was a generative way of representing the data where we tried to model the world, and CRF, which was a discriminative method, where we were interested in specific features that distinguished objects without bothering trying to model the world. There is a dichotomy between generative and discriminative approaches. We will use a Bayesian approach to classify mitochondrial proteins, and SVM to classify tumor samples.

In this lecture we will look at two new algorithms: a generative classifier, Nave Bayes, and a discriminative classifier, Support Vector Machines (SVMs). We will discuss biological applications of each of these models, specifically in the use of Nave Bayes classifiers to predict mitochondrial proteins across the genome and the use of SVMs for the classification of cancer based on gene expression monitoring by DNA microarrays. The salient features of both techniques and caveats of using each technique will also be discussed.

14.2 Classification - Bayesian Techniques

Consider the problem of identifying mitochondrial proteins. If we look at the human genome, how do we determine which proteins are involved in mitochondrial processes, or more generally which proteins are targeted to the mitochondria? This is particularly useful because if we know the mitochondrial proteins, we can study how these proteins mediate disease processes and metabolic functions. The classification method we will look considers 7 features for all human proteins:

1. targeting signal
2. protein domains
3. co-expression
4. mass spectrometry
5. sequence homology
6. induction
7. motifs

Our overall approach will be to determine how these features are distributed for both mitochondrial and non-mitochondrial proteins. Then, given a new protein, we can apply probabilistic analysis to these seven features to decide which class it most likely falls into.

14.2.1 Single Features and Bayes Rule

Let's just focus on one feature at first. We must first assume that there is a class dependent distribution for the features. We must first derive this distribution from real data. The second thing we need is the a priori chance of drawing a sample of particular class before looking at the data. The chance of getting a particular class is simply the relative size of the class. Once we have these probabilities, we can use Bayes

---

1 Mitochondria is the energy producing machinery of cell. Very early in life, the mitochondria was engulfed by the predecessor to modern day eukaryotes, and now, we have different compartments in our cells. So the mitochondria has its own genome, but it is very depleted from its own ancestral genome - only about 11 genes remain. But there are hundreds are genes that make the mitochondria work, and these proteins are encoded by genes transcribed in the nucleus, and then transported to the mitochondria. So the goal is to figure out which proteins encoded in the genome are targeted to the mitochondria. This is important because there are many diseases associated with the mitochondria, such as aging.
rule to get the probability a sample is in a particular class given the data (this is called the posterior). We have forward generative probabilities, and use Bayes rules to perform the backwards inference. Note that it is not enough to just consider the probability the feature was drawn from each class dependent distribution, because if we knew a priori that one class (say class A) is much more common than the other, then it should take overwhelming evidence that the feature was drawn from class B’s distribution for us to believe the feature was indeed from class B. The correct way to find what we need based on both evidence and prior knowledge is to use Bayes Rule:

\[
P(\text{Class}|\text{feature}) = \frac{P(\text{feature}|\text{Class})P(\text{Class})}{P(\text{feature})}
\]

- **Posterior**: \(P(\text{Class}|\text{feature})\)
- **Prior**: \(P(\text{Class})\)
- **Likelihood**: \(P(\text{feature}|\text{Class})\)

This formula gives us exactly the connection we need to flip known feature probabilities into class probabilities for our classifying algorithm. It lets us integrate both the likelihood we derive from our observations and our prior knowledge about how common something is. In the case of mtDNA, for example, we can estimate that mitochondrial DNA makes up something less than 10% of the human genome. Therefore, applying Bayes rule, our classifier should only classify a gene as mitochondrial if there is a very strong likelihood based on the observed features, since the prior probability that any gene is mitochondrial is so low.

With this rule, we can now form a maximum likelihood rule for predicting an object’s class based on an observed feature. We want to choose the class that has the highest probability given the observed feature, so we will choose Class1 instead of Class2 if:

\[
\frac{P(\text{feature}|\text{Class1})P(\text{Class1})}{P(\text{feature})} > \frac{P(\text{feature}|\text{Class2})P(\text{Class2})}{P(\text{feature})}
\]

Notice that \(P(\text{feature})\) appears on both sides, so we can cancel that out entirely, and simply choose the class with the highest value of \(P(\text{feature}|\text{Class})P(\text{Class})\).

Another way of looking at this is as a discriminant function: By rearranging the formulas above and taking the logarithm, we should select Class1 instead of Class2 precisely when

\[
\log \left( \frac{P(\text{feature}|\text{Class1})P(\text{Class1})}{P(\text{feature}|\text{Class2})P(\text{Class2})} \right) > 0
\]

In this case the use of logarithms provide distinct advantages:

1. Numerical stability
2. Easier math (its easier to add the expanded terms than multiply them)
3. Monotonically increasing discriminators.

This discriminant function does not capture the penalties associated with misclassification (in other words, is one classification more detrimental than other). In this case, we are essentially minimizing the number of misclassifications we make overall, but not assigning penalties to individual misclassifications. From examples discussed in class and in the problem set - if we are trying to classify a patient as having cancer or not, it could be argued that it is far more harmful to misclassify a patient as being healthy if they have cancer than to misclassify a patient as having cancer if they are healthy. In the first case, the patient will not be treated and would be more likely to die, whereas the second mistake involves emotional grief but no greater chance of loss of life. To formalize the penalty of misclassification we define something called a loss function, \(L_{kj}\), which assigns a loss to the misclassification of an object as class j when the true class is class k (a specific example of a loss function was seen in Problem Set 2).

### 14.2.2 Collecting Data

The preceding tells us how to handle predictions if we already know the exact probabilities corresponding to each class. If we want to classify mitochondrial proteins based on feature \(X\), we still need ways of determining the probabilities \(P(\text{mito}), P(\text{mito}), P(X|\text{mito})\) and \(P(X|\text{mito})\). To do this, we need a training set: a set of data that is already classified that our algorithm can use to learn the distributions corresponding to each class. A high-quality training set (one that is both large and unbiased) is the most important part of any
classifier. An important question at this point is, how much data do we need about known genes in order to build a good classifier for unknown genes? This is a hard question whose answer is not fully known. However, there are some simple methods that can give us a good estimate: when we have a fixed set of training data, we can keep a holdout set that we dont use for our algorithm, and instead use those (known) data points to test the accuracy of our algorithm when we try to classify them. By trying different sizes of training versus holdout set, we can check the accuracy curve of our algorithm. Generally speaking, we have enough training data when we see the accuracy curve flatten out as we increase the amount of training data (this indicates that additional data is likely to give only a slight marginal improvement). The holdout set is also called the test set, because it allows us to test the generalization power of our classifier.

Supposing we have already collected our training data, however, how should we model \( P(X|\text{Class}) \)? There are many possibilities. One is to use the same approach we did with clustering in the last lecture and model the feature as a Gaussian then we can follow the maximum likelihood principle to find the best center and variance. The one used in the mitochondrial study is a simple density estimate: for each feature, divide the range of possibilities into a set of bins (say, five bins per feature). Then we use the given data to estimate the probability of a feature falling into each bin for a given class. The principle behind this is again maximum likelihood, but for a multinomial distribution rather than a Gaussian. We may choose to discretize a otherwise continuous distribution because estimating a continuous distribution can be complex.

There is one issue with this strategy: what if one of the bins has zero samples in it? A probability of zero will override everything else in our formulas, so that instead of thinking this bin is merely unlikely, our classifier will believe it is impossible. There are many possible solutions, but the one taken here is to apply the Laplace Correction: add some small amount (say, one element) into each bin, to draw probability estimates slightly towards uniform and account for the fact that (in most cases) none of the bins are truly impossible. Another way to avoid having to apply the correction is to choose bins that are not too small so that bins will not have zero samples in them in practice. If you have many many points, you can have more bins, but run the risk of overfitting your training data.

14.2.3 Estimating Priors

We now have a method for approximating the feature distribution for a given class, but we still need to know the relative probability of the classes themselves. There are three general approaches:

1. Estimate the priors by counting the relative frequency of each class in the training data. This is prone to bias, however, since data available is often skewed disproportionately towards less common classes (since those are often targeted for special study). If we have a high-quality (representative) sample for our training data, however, this works very well.

2. Estimate from expert knowledge there may be previous estimates obtained by other methods independent of our training data, which we can then use as a first approximation in our own predictions. In other words, you might ask experts what the percentage of mitochondrial proteins are.

3. Assume all classes are equally likely we would typically do this if we have no information at all about the true frequencies. This is effectively what we do when we use the maximum likelihood principle: our clustering algorithm was essentially using Bayesian analysis under the assumption that all priors are equal. This is actually a strong assumption, but when you have no other data, this is the best you can do.

For classifying mitochondrial DNA, we use method (2), since some estimates on the proportions of mtDNA were already known. But there is an complication there are more than 1 features.

14.2.4 Multiple features and Nave Bayes

In classifying mitochondrial DNA, we were looking at 7 features and not just one. In order to use the preceding methods with multiple features, we would need not just one bin for each individual feature range, but one for each combination of features if we look at two features with five ranges each, thats already 25 bins. All seven features gives us almost 80,000 bins and we can expect that most of those bins will be empty
simply because we don’t have enough training data to fill them all. This would cause problems because zeros cause infinite changes in the probabilities of being in one class. Clearly this approach won’t scale well as we add more features, so we need to estimate combined probabilities in a better way.

The solution we will use is to assume the features are independent, that is, that once we know the class, the probability distribution of any feature is unaffected by the values of the other features. This is the Naive Bayes Assumption, and it is almost always false, but it is often used anyway for the combined reasons that it is very easy to manipulate mathematically and it is often close enough to the truth that it gives a reasonable approximation. (Note that this assumption does not say that all features are independent: if we look at the overall model, there can be strong connections between different features, but the assumption says that those connections are divided by the different classes, and that within each individual class there are no further dependencies.) Also, if you know that some features are coupled, you could learn the joint distribution in only some pairs of the features.

Once we assume independence, the probability of combined features is simply the product of the individual probabilities associated with each feature. So we now have:

\[ P(f_1, f_2, K, f_N | \text{Class}) = P(f_1 | \text{Class})P(f_2 | \text{Class})KP(f_N | \text{Class}) \]

Where \( f_1 \) represents feature 1. Similarly, the discriminant function can be changed to the multiplication of the prior probabilities:

\[ G(f_1, f_2, K, f_N) = \log \left( \frac{\Pi P(f_1 | \text{Class}_1)P(\text{Class}_1)}{\Pi P(f_1 | \text{Class}_2)P(\text{Class}_2)} \right) \]

### 14.2.5 Testing a classifier

A classifier should always be tested on data not contained in its training set. We can imagine in the worst case an algorithm that just memorized its training data and behaved randomly on anything else a classifier that did this would perform perfectly on its training data, but that indicates nothing about its real performance on new inputs. This is why it’s important to use a test, or holdout, set as mentioned earlier. However, a simple error rate doesn’t encapsulate all of the possible consequences of an error. For a simple binary classifier (an object is either in or not in a single target class), there are the following for types of errors:

1. True positive (TP)
2. True negative (TN)
3. False positive (FP)
4. False negative (FN)

The frequency of these errors can be encapsulated in performance metrics of a classifier which are defined as,

1. **Sensitivity** what fraction of objects that are in a class are correctly labeled as that class? That is, what fraction have true positive results? High sensitivity means that elements of a class are very likely to be labeled as that class. Low sensitivity means there are too many false negatives.

2. **Specificity** what fraction of objects not in a class are correctly labeled as not being in that class? That is, what fraction have true negative results? High specificity means that elements labeled as belonging to a class are very likely to actually belong to it. Low specificity means there are too many false positives.

In most algorithms there is a tradeoff between sensitivity and specificity. For example, we can reach a sensitivity of 100% by labeling everything as belonging to the target class, but we will have a specificity of 0%, so this is not useful. Generally, most algorithms have some probability cutoff they use to decide whether to label an object as belonging to a class (for example, our discriminant function above). Raising that threshold increases the specificity but decreases the sensitivity, and decreasing the threshold does the reverse. The MAESTRO algorithm for classifying mitochondrial proteins (described in this lecture) achieves 99% specificity and 71% sensitivity.
14.2.6 MAESTRO Mitochondrial Protein Classification

They find a class dependent distribution for each features by creating several bins and evaluating the proportion of mitochondrial and non mitochondrial proteins in each bin. This lets you evaluate the usefulness of each feature in classification. You end up with a bunch of medium strength classifiers, but when you combine them together, you hopefully end up with a stronger classifier. Calvo et al. (2006)TODO: insert BibTex citation @scribe; sought to construct high-quality predictions of human proteins localized to the mitochondrion by generating and integrating data sets that provide complementary clues about mitochondrial localization. Specifically, for each human gene product p, they assign a score s_i(p), using each of the following seven genome-scale data sets targeting signal score, protein domain score, cis-motif score, yeast homology score, ancestry score, coexpression score, and induction score (details of each of the meaning and content of each of these data sets can be found in the manuscript). Each of these scores s_1 − S_7 can be used individually as a weak genome-wide predictor of mitochondrial localization. Each methods performance was assessed using large gold standard curated training sets - 654 mitochondrial proteins T_mito maintained by the MitoP2 database and 2,847 nonmitochondrial proteins T_mito annotated to localize to other cellular compartments. To improve prediction accuracy, the authors integrated these eight approaches using a nave Bayes classifier that was implemented as a program called MAESTRO. So we can take several weak classifiers, and combine them to get a stronger classifier.

When MAESTRO was applied across the human proteome, 1451 proteins were predicted as mitochondrial proteins and 450 novel proteins predictions were made. As mentioned in the previous section The MAESTRO algorithm achieves a 99% specificity and a 71% sensitivity for the classification of mitochondrial proteins, suggesting that even with the assumption of feature independence, Nave Bayes classification techniques can prove extremely powerful for large-scale (i.e. genome-wide) scale classification.

14.3 Classification Support Vector Machines

The previous section looked at using probabilistic (or generative) models for classification, this section looks at using discriminative techniques in essence, can we run our data through a function to determine its structure? Such discriminative techniques avoid the inherent cost involved in generative models which might require more information than is actually necessary.

Support vector machine techniques essentially involve drawing a vector that’s perpendicular to the line (hyperplane) separating the training data. The approach is that we look at the training data to obtain a separating hyperplane so that two classes of data lie on different sides of the hyperplane. There are, in general, many hyperplanes that can separate the data, so we want to draw the hyperplane that separates the data the most - we wish to choose the line that maximizes the distance from the hyperplane to any data point. In other words, the SVM is a maximum margin classifier. You can think of the hyperplane being surrounded with margins of equal size on each side of the line, with no data points inside the margin on either side. We want to draw the line that allows us to draw the largest margin. Note that once the separating line and margin are determined, some data points will be right on the boundary of the margin. These are the data points that keep us from expanding the margin any further, and thus determine the line/margin. Such points are called the support vectors. If we add new data points outside the margin or remove points that are not support vectors, we will not change the maximum margin we can achieve with any hyperplane.

Suppose that the vector perpendicular to the hyperplane is w, and that the hyperplane passes through the point \( \left( \frac{b}{||w||} \right) \). Then a point \( x \) is classified as being in the positive class if \( w \cdot x \) is greater than \( b \), and negative otherwise. It can be shown that the optimal w, that is, the hyperplane that achieves the maximum margin, can actually be written as a linear combination of the data vectors \( \sum a_i x_i \). Then, to classify a new data point \( x \), we need to take the dot product of w with \( x \) to arrive at a scalar. Notice that this scalar, \( \sum a_i (x_i \cdot x) \) only depends on the dot product between x and the training vectors \( x_i \). Furthermore, it can be shown that finding the maximum margin hyperplane for a set of (training) points amounts to maximizing a linear program where the objective function only depends on the dot product of the training points with each other. This is good because it tells us that the complexity of solving that linear program is independent of the of dimension of the data points. If we precompute the pairwise dot products of the training vectors, then it makes no difference what the dimensionality of the data is in regards to the running time of solving.
the linear program.

14.3.1 Kernels

We see that SVMs are dependent only on the dot product of the vectors. So, if we call our transformation \( \phi(v) \), for two vectors we only care about the value of \( \phi(v_1) \cdot \phi(v_2) \) The trick to using kernels is to realize that for certain transformations \( \phi \), there exists a function \( K(v_1, v_2) \), such that:

\[
K(v_1, v_2) = \phi(v_1) \cdot \phi(v_2)
\]

In the above relation, the right-hand side is the dot product of vectors with very high dimension, but the left-hand side is the function of two vectors with lower dimension. In our previous example of mapping \( x \rightarrow (x, y = x^2) \), we get

\[
K(x_1, x_2) = (x_1 x_2^2) \cdot (x_2, x_2^2) = x_1 x_2 + (x_1 x_2)^2
\]

Now we did not actually apply the transformation \( \phi \), we can do all our calculations in the lower dimensional space, but get all the power of using a higher dimension.

Example kernels are the following:

1. Linear kernel: \( K(v_1, v_2) = v_1 \cdot v_2 \) which represents the trivial mapping of \( \phi(x) = x \)
2. Polynomial kernel: \( K(v_1, v_2) = (1 + v_1 \cdot v_2)^n \) which was used in the previous example with \( n = 2 \).
3. Radial basis kernel: \( K(v_1, v_2) = \exp(-\beta|v_1 - v_2|^2) \) This transformation is actually from a point \( v_1 \) to a function (which can be thought of as being a point in Hilbert space) in an infinite-dimensional space. So what were actually doing is transforming our training set into functions, and combining the to get a decision boundary. The functions are Gaussians centered at the input points.
4. Sigmoid kernel: \( K(v_1, v_2) = \tanh[\beta(v_1^T v_2 + r)] \) Sigmoid kernels have been popular for use in SVMs due to their origin in neural networks (e.g. sigmoid kernel functions are equivalent to two-level, perceptron neural networks). It has been pointed out in previous work (Vapnik 1995) that the kernel matrix may not be positive semi-definite for certain values of the parameters \( \mu \) and \( r \). The sigmoid kernel has nevertheless been used in practical applications (see Scholkopf 1997) TODO: insert BibTex citation.

Here is a specific example of a kernel function. Consider the two classes of one-dimensional data:

\{\{-5,-4,-3,2,4,5\} and \{-2,-1,0,1,2\} \}

This data is clearly not linearly separable, and the best separation boundary we can find might be \( x > 2.5 \). Now consider applying the transformation \( \phi \). The data can now be written as new pairs,

\{\{-5,-4,-3,2,4,5\} \rightarrow \{(−5,25),(−4,16),(−3,9),(3,9),(4,16),(5,25)\} \}

and

\{-2,-1,0,1,2\} \rightarrow \{(-2,−4),(-1,1),(0,0),(1,1),(2,4)\}

This data is separable by the rule \( y > 6.5 \), and in general the more dimensions we transform data to the more separable it becomes.

An alternate way of thinking of this problem is to transform the classifier back in to the original low-dimensional space. In this particular example, we would get the rule \( x^2 < 6.5 \), which would bisect the number line at two points. In general, the higher dimensionality of the space that we transform to, the more complicated a classifier we get when we transform back to the original space.

One of the caveats of transforming the input data using a kernel is the risk of overfitting (or overclassifying) the data. More generally, the SVM may generate so many feature vector dimensions that it does not generalize well to other data. To avoid overfitting, cross-validation is typically used to evaluate the fitting provided by each parameter set tried during the grid or pattern search process. In the radial-basis kernel, you can essentially increase the value of \( \beta \) until each point is within its own classification region (thereby defeating the classification process altogether). SVMs generally avoid this problem of over-fitting due to the fact that they maximize margins between data points.

When using difficult-to-separate training sets, SVMs can incorporate a cost parameter \( C \), to allow some flexibility in separating the categories. This parameter controls the trade-off between allowing training errors and forcing rigid margins. It can thereby create a soft margin that permits some misclassifications.
Increasing the value of $C$ increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well.

Can we use just any function as our kernel? The answer to this is provided by Mercers Condition which provides us an analytical criterion for choosing an acceptable kernel. Mercers Condition states that a kernel $K(x,y)$ is a valid kernel if and only if the following holds (Burgess 1998):

\[ \int K(x,y)g(x)g(y)\,dx\,dy \geq 0 \]

In all, we have defined SVM discriminators and shown how to perform classification with appropriate kernel mapping functions that allow performing computations on lower dimension while being to capture all the information available at higher dimensions. The next section describes the application of SVMs to the classification of tumors for cancer diagnostics.

### 14.4 Tumor Classification with SVMs

A generic approach for classifying two types of acute leukemias acute myeloid leukemia (AML) and acute lymphoid leukemia (ALL) was presented by Golub et al. (1999). This approach centered on effectively addressing three main issues:

1. Whether there were genes whose expression pattern to be predicted was strongly correlated with the class distinction (i.e. can ALL and AML be distinguished)

2. How to use a collection of known samples to create a class predictor capable of assigning a new sample to one of two classes

3. How to test the validity of their class predictors

They addressed (1) by using a neighbourhood analysis technique to establish whether the observed correlations were stronger than would be expected by chance. This analysis showed that roughly 1100 genes were more highly correlated with the AML-ALL class distinction than would be expected by chance. To address (2) they developed a procedure that uses a fixed subset of informative genes (chosen based on their correlation with the class distinction of AML and ALL) and makes a prediction based on the expression level of these genes in a new sample. Each informative gene casts a weighted vote for one of the classes, with the weight of each vote dependent on the expression level in the new sample and the degree of that genes correlation with the class distinction. The votes are summed to determine the winning class. To address (3) and effectively test their predictor by first testing by cross-validation on the initial data set and then assessing its accuracy on an independent set of samples. Based on their tests, they were able to identify 36 of the 38 samples (which were part of their training set!) and all 36 predictions were clinically correct. On the independent test set 29 of 34 samples were strongly predicted with 100% accuracy and 5 were not predicted.

A SVM approach to this same classification problem was implemented by Mukherjee et al. (2001). The output of classical SVM is a binary class designation. In this particular application it is particularly important to be able to reject points for which the classifier is not confident enough. Therefore, the authors introduced a confidence interval on the output of the SVM that allows for rejection of points with low confidence values. As in the case of Golub et al., it was important for the authors to infer which genes are important for the classification. The SVM was trained on the 38 samples in the training set and tested on the 34 samples in the independent test set (exactly in the case of Golub et al.). The authors results are summarized in the following table (where $|d|$ corresponds to the cutoff for rejection).

| Genes | Rejects | Errors | Confidence level | $|d|$ |
|-------|---------|--------|-----------------|------|
| 7129  | 3       | 0      | 93%             | 0.1  |
| 40    | 0       | 0      | 93%             | 0.1  |
| 5     | 3       | 0      | 92%             | 0.1  |

These results a significant improvement over previously reported techniques, suggesting that SVMs play an important role in classification of large data sets (as those generated by DNA microarray experiments).
14.5 Semi-Supervised Learning

In some scenarios we have a data set with only a few labeled data points, a large number of unlabeled data points and inherent structure in the data. This type of scenario both clustering and classification do not perform well and a hybrid approach is required. This semi-supervised approach could involve the clustering of data first followed by the classification of the generated clusters.

14.5.1 History of the Field

14.5.2 Open Problems

14.6 Current Research Directions

14.7 Further Reading

14.8 Tools and Techniques

14.9 What Have We Learned?

Bibliography
CHAPTER

FIFTEEN

REGULATORY MOTIFS, GIBBS SAMPLING, AND EM

Bianca Dumitrascu and Neal Wadhwa (2010)
Joseph Lane (2009)
Brad Cater and Ethan Heilman (2008)

List of figures

Transcription Factors Binding at Motif Sites
Example Profile Matrix
Z Matrix Found by EM, Gibbs Sampling, & Greedy Algorithm
Comparison of EM, Gibbs Sampling, & Greedy Approaches
Sample Position Weight Matrix
Gibbs Sampling by AlignACE and BioProspector
Sequences with zero, one or two motifs
Coin Toss Example
Motif Logo
Motif Logo KL Distance
15.1 Introduction

We have already explored the areas of dynamic programming, sequence alignment, sequence classification and modeling, hidden Markov models, and expectation maximization. As we will see, all of these techniques will be useful in both discovering new motifs and investigating their functions.

15.1.1 What are Motifs?

Motifs are short (6-8 bases long), recurring patterns in DNA that are believed to have a biological function. They can, for instance, indicate sequence-specific binding sites for proteins such as nucleases and transcription factors. As we have discussed, genes are turned on or off in response to changing environments. Regulatory motifs are the correct sequence tags which specialized proteins, called transcription factors, recognize and to which they bind. Once bound, the transcription factors enable the expression of a gene. Transcription factors often regulate a group of genes that are involved in a similar cellular function. Thus, genes that contain the same motif in their upstream regions are likely to be related in their functions. In fact, many regulatory motifs are found through analysis of the upstream regions of genes that are known to have similar functions.

Motifs have become exceedingly useful for defining genetic regulatory networks and deciphering the functions of individual genes. With our current computational abilities, regulatory motif discovery and analysis has progressed considerably and remains at the forefront of genomic studies.

15.1.2 What is the Role of Transcription Factors?

Transcription factors (TFs) are proteins that bind to specific DNA sequences, in this case they are able to recognize regulatory motifs. Proteins can 'feel' DNA by reading the chemical properties of the bases without opening the DNA. Depending on the 3D topology, TFs can upregulate or downregulate gene expression. TFs use several mechanisms in order to control gene expression, among these are acetylation of histone proteins, deacetylation of histone proteins, recruitment of co-repressor or coactivator proteins to the transcription factor DNA complex, stabilization or blocking of RNA binding to DNA.

Other types of recognition are given by microRNAs, which bind to motifs given strict conditions of complementarity (generally first 7 bases are those that bind to the motif), Nucleosomes that recognize motifs based on their GC content, and even RNAs.

15.2 Problems in Motif Discovery

Before we can get into algorithms for motif discovery, we must first understand the characteristics of motifs, especially those that make motifs somewhat difficult to find. As mentioned above, motifs are generally very short, usually only 6-8 base pairs long. Additionally, motifs can be degenerate, that is nucleotides at locations along the motif can be swapped out for different nucleotides without affecting the motif’s function. Motifs can be degenerate because instead of recognizing bases at a complementary level, the proteins recognize the base pairs by 'feel', or their chemical properties. As seen in ??, a protein may only care about a portion of the motif (sides or center) and thus part of the motif may not matter. The parts that matter depends on the physical structure of the protein. A protein may only be sensitive to the presence of a purine and thus be unable to distinguish between A and G. The topology of these proteins matters greatly. The location of every atom matters and determines specificity in a way that is very hard to predict given the protein’s structure.

This creates a challenging problem. If we were only looking for a fixed k-mer, we could simply search for the k-mer in all the sequences we are looking at using local alignment tools. However, the motif may be different in different sequences. Because of this, a string of nucleotides that is known to be a regulatory motif is said to be an instance of a motif because it represents one fo potentially many different combinations of nucleotides that fulfill the function of the motif.

In our approaches, we make two assumptions about the data. The first is that there are no pairwise correlations between cases. We assume that each base is independent of every other base. This is true much of the time, but these correlations do exist in real life. if we took them into account, the number of
parameters would increase exponentially and we would run the risk of overfitting our data. The second assumption we make is that all motifs have fixed lengths. However, this is not necessarily the case. But such an assumption simplifies the problem greatly. Despite this, motif finding is still a very challenging problem. Motifs can contain any set of nucleotides and are very short, making it difficult to locate them. A motif can occur either upstream or downstream from the target gene and be located at varying distances, sometimes up to $10^k$ to $10^M$ base pairs, from the gene’s start.

Because motif instances can vary greatly, we end up looking for a profile matrix to characterize the motif. This matrix gives us the frequency of each base at each location in the motif. In the adjacent figure, there is an example profile matrix. We also want to compute the probability that the $i$th character in each sequence is the start of a motif. Once we have these probabilities, we can take the maximum probabilities to point to specific motif instances. These two things are both unknown and we are going to try to estimate them simultaneously. We denote the profile matrix by $p_{ik}$ where $k$ corresponds to the position within the motif. If $k = 0$, then this matrix will give us the background distribution.

We now define the problem of motif finding more rigorously. We assume that we are given a set of co-regulated and functionally related genes. In the past, many motifs were discovered by doing footprint experiments. One would look to see where the transcription factors were binding and isolate that part of the sequence. Since motifs are highly involved in the regulation of protein building, these regions were likely to correspond to motifs. There are several computational methods that we could use to locate a motif:

1. Perform a local alignment across the set of sequences and explore the alignments that resulted in a very high alignment score.

2. Model the promoter regions using an HMM and then use a generative model to find non-random sequences.

3. Reduce the search space by applying expert knowledge for what motifs should look like.

4. Search for conserved blocks from two different sequences.
5. If there is a specific region that is highly likely to contain a motif, count the words that occur in each region.

6. Use probabilistic methods, such as EM, Gibbs Sampling, or a greedy algorithm

Considering option 5, there are a few difficulties that must be considered. For example, there could be many common words that occur in these regions that are in fact not a regulatory motif but a different set of instructions. Furthermore, given a list of words that could be a motif, one could either choose the most common word or the word that appears least often. In fact, motifs are generally over represented in promoter regions. On the other hand, if there are too many of them, transcription factors may be unable to bind. One possible approach would be to find the sequence that is most common in promoter regions but that is less likely to occur in other regions. This strategy is commonly performed as a post processing step to narrow down the number of possible motifs.

In 2003, Professor Kellis argued that there must be some selective pressure to causes a particular sequence to be occur on specific places. His PhD. thesis on the topic can be found at the following location: http://web.mit.edu/manoli/www/publications/Kellis_Thesis_03.pdf.

In the next section, we will talk more about EM, Gibbs Sampling, and the Greedy Algorithm. These algorithms are the focus of this lecture.

### 15.3 Computational Methods of Finding Motifs

In EM, Gibbs Sampling and the Greedy Algorithm, we first compute a profile matrix and then use this to compute the matrix $Z_{ij}$, which is the probability that there is a motif instance at position $j$ in the sequence $i$. This Z matrix is then used to recompute the original profile matrix until convergence. Some examples of this matrix are graphically represented by ??

Intuitively, the greedy algorithm will always pick the most probable location for the motif. The EM
algorithm will take an average while Gibbs Sampling will actually use the probability distribution given by $Z$ to sample a motif in a step.

Figure 15.4: Selecting motif location: the greedy algorithm will always pick the most probable location for the motif. The EM algorithm will take an average while Gibbs Sampling will actually use the probability distribution given by $Z$ to sample a motif in each step

15.3.1 Expectation Maximization

Step 1: Initialization The first step in EM is to generate an initial probability weight matrix (PWM). The PWM describes the frequency of each nucleotide at each location in the motif. In ??, there is an example of a PWM. In this example, we assume that the motif is eight bases long.

If you are given a set of aligned sequences and the location of suspected motifs within them, then finding the PWM is accomplished by counting the number of bases in each position in every suspected motif. We can initialize PWM by choosing starting locations randomly. It is important to realize that the result of the EM is entirely dependent on the initial parameters, that is the initial value of the PWM. It is good practice to run the EM algorithm multiple times using different initial values. This helps ensure that the solution we find is not only locally optimal, but also globally optimal.

We refer to the PWM as $p_{ck}$, where $p_{ck}$ is the probability of character $c$ occurring in position $k$ of the motif. Note: if there is 0 probability, it is generally a good idea to insert pseudo-counts into your probabilities. The PWM is also called the profile matrix. In addition to the PWM, we also keep a distribution of the bases not in the motif.

![sequence positions](image)

**Figure 15.5: Sample position weight matrix**

Step 2: Expectation In the expectation step, we generate a vector $Z_{ij}$ which contains the probability of the motif starting in position $j$ in sequence $i$. In EM, the $Z$ vector gives us a way of classifying all of the nucleotides in the sequences and tell us whether they are part of the motif or not. We can calculate $Z_{ij}$ using Bayes' Rule. This simplifies to:

$$Z_{ij}^t = \frac{Pr(X_i | Z_{ij} = 1, p^t)Pr(Z_{ij} = 1)}{\sum_{k=1}^{L-1} \sum_{p=1}^{m} Pr(X_i | Z_{ij} = 1, p^t)Pr(Z_{ik} = 1)}$$

where $Pr(X_i | Z_{ij} = 1, p^t)$ is defined as
6.047/6.878 Lecture 15: Regulatory Motifs, Gibbs Sampling, and EM

This probability is the probability of sequence \( i \) existing given that the motif is in the position \( j \) in the sequence. The first and last product correspond to the probability that most of the sequence comes from some background probability distribution while the middle product corresponds to the motif coming from a motif probability distribution. In this equation, we assume that the sequence has length \( L \) and the motif has length \( W \).

**Step 3: Maximization** Once we have calculated \( Z \), we can use the results to update both the PWM and the background probability distribution. We can update the PWM using the following equation:

\[
A_{ij} = \frac{\prod_{k=j}^{j+W-1} p_{ck}, k-j+1 \prod_{k=j}^{j+W-1} p_{ck}, 0}{\prod_{k=j}^{j+W-1} p_{ck}, k-j}
\]

**Step 4: Repeat** Repeat steps 2 and 3 until convergence.

One possible way to test whether the profile matrix has converged is to measure how much each element in the PWM changes after step maximization. If the change is below a chosen threshold, then we can terminate the algorithm. It is advisable to rerun the algorithm with different initial starting positions to try and reduce the chance of converging on a local maximum that is not the global maximum and to get a good sense of the solutions space.

### 15.3.2 Gibbs Sampling

Gibbs sampling is similar to EM except that it is a stochastic process, while EM is deterministic. In the expectation step, we only consider nucleotides within the motif window in Gibbs sampling. In the maximization step, we sample from \( Z_{ij} \) and use the result to update the PWM instead of averaging over all values as in EM.

**Step 1: Initialization** As with EM, you generate your initial PWM based off of a random sampling of initial starting positions. Unlike EM, the initial PWM does not determine the algorithm's outcome. This is because Gibbs sampling updates its PWM based off of a random sample. An advantage of this is that the algorithm is more robust to getting stuck in a local maximum.

**Step 2: Remove** Remove one sequence, \( x \), from your set of sequences. You will change the starting location of for this particular sequence.

**Step 3: Update** Using the remaining set of sequences, update the PWM by counting how often each base occurs in each position.
Step 4: Sample Using the newly updated PWM, generated a probability distribution for starting positions in sequence x. To generate each element, Ai,j, of the probability distribution, the following formula is used,

\[ A_{ij} = \frac{\prod_{k=1}^{n} p_{ik,k-j+1}}{\prod_{k=1}^{n} p_{i0,0}} \]

This is simply the probability that the sequence was generated using our motif PWM divided by the probability that the sequence was generated using our background PWM.

Select a new starting position for x by randomly sampling from this distribution.

Step 5: Iterate Loop back to Step 2 and iterate the algorithm until convergence.

Two popular implementations of Gibbs Sampling applied to this problem are AlignACE and BioProspector. A more general Gibbs Sampler can be found in the program WinBUGS. Both AlignACE and BioProspector use the aforementioned algorithm for several choices of initial values and then report common motifs. Gibbs sampling is easier to implement than E-M and in theory, it is less likely to get stuck at a local optimum. However, the search is less systematic.

15.3.3 Greedy

While the greedy algorithm is not used very much in practice, it is important know how it functions and mainly its advantages and disadvantages compared to EM and Gibbs sampling. The Greedy algorithm works just like Gibbs sampling except for a main difference in Step 4. Instead of randomly choosing selecting a new starting location, it simply picks the starting location with the highest probability.

This makes the Greedy algorithm quicker, although only slightly, than Gibbs sampling but reduces its chances of finding a global maximum considerably. In cases where the starting location probability distribution is fairly level, the greedy algorithm ignores the weights of every other starting position other than the most likely.

15.4 OOPS,ZOOPS,TCM

The different types of sequence model make differing assumptions about how and where motif occurrences appear in the dataset. The simplest model type is OOPS (One-Occurrence-Per-Sequence) since it assumes that there is exactly one occurrence per sequence of the motif in the dataset. This is the case we have analyzed in the previous section. This type of model was introduced by Lawrence & Reilly (1990) TODO:
6.047/6.878 Lecture 15:
Regulatory Motifs, Gibbs Sampling, and EM

REFERENCE NEEDED! @scribe: , when they describe for the first time a generalization of OOPS, called ZOOPS (Zero-or-One-Occurrence-Per-Sequence), which assumes zero or one motif occurrences per dataset sequence. Finally, TCM (Two-Component Mixture) models assume that there are zero or more non-overlapping occurrences of the motif in each sequence in the dataset, as described by Baily & Elkan (1994). TODO: REFERENCE NEEDED! @scribe: Each of these types of sequence model consists of two components, which model, respectively, the motif and non-motif (background) positions in sequences. A motif is modelled by a sequence of discrete random variables whose parameters give the probabilities of each of the different letters (4 in the case of DNA, 20 in the case of proteins) occurring in each of the different positions in an occurrence of the motif. The background positions in the sequence are modelled by a single discrete random variable.

15.5 Extension of the EM Approach
15.5.1 ZOOPS Model
The approach presented before (OOPS) relies on the assumption that every sequence is characterized by only one motif (e.g., there is exactly one motif occurrence in a given sequence). The ZOOPS model takes into consideration the possibility of sequences not containing motifs.

In this case let \( i \) be a sequence that does not contain a motif. This extra information is added to our previous model using another parameter \( \lambda \) to denote the prior probability that any position in a sequence is the start of a motif. Next, the probability of the entire sequence to contain a motif is \( \lambda = (L - W + 1) * \lambda \)

The E-Step
The E-step of the ZOOPS model calculates the expected value of the missing information—the probability that a motif occurrence starts in position \( j \) of sequence \( X_i \). The formulas used for the three types of model are given below.

\[
Z_{ij}^{(t)} = \frac{\Pr(X_i | Z_{ij} = 1, p^{(t)}) \lambda^{(t)}}{\Pr(X_i | Q_i = 0, p^{(t)}) (1 - \gamma^{(t)}) + \sum_{k=1}^{L-W+1} \Pr(X_i | Z_{ik} = 1, p^{(t)}) \lambda^{(t)}}
\]

The M-Step
The M-step of EM in MEME re-estimates the values for \( \lambda \) using the preceding formulas. The math remains the same as for OOPS, we just update the values for \( \lambda \) and \( \gamma \)

\[
\lambda^{(t+1)} = \frac{\gamma^{(t+1)} (L - W + 1)}{n (L - W + 1)} = \frac{1}{n (L - W + 1)} \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)}
\]

The model above takes into consideration sequences that do not have any motifs. The challenge is to also take into consideration the situation in which there is more than one motif per sequence occurrence. This can be accomplished with the more general model TCM. TCM (two-component mixture model) is based on the assumption that there can be zero, one, or even two motif occurrences per sequence.

15.5.2 Finding Multiple Motifs
All the above sequence model types model sequences containing a single motif (notice that TCM model can describe sequences with multiple occurrences of the same motif). To find multiple, non-overlapping, different motifs in a single dataset, one incorporates information about the motifs already discovered into
the current model to avoid rediscovering the same motif. The three sequence model types assume that motif occurrences are equally likely at each position $j$ in sequences $x_i$. This translates into a uniform prior probability distribution on the missing data variables $Z_{ij}$. A new prior on each $Z_{ij}$ had to be used during the E-step that takes into account the probability that a new width-$W$ motif occurrence starting at position $X_{ij}$ might overlap occurrences of the motifs previously found. To help compute the new prior on $Z_{ij}$ we introduce variables $V_{ij}$ where $V_{ij} = 1$ if a width-$W$ motif occurrence could start at position $j$ in the sequence $X_i$ without overlapping an occurrence of a motif found on a previous pass. Otherwise $V_{ij} = 0$.

$$V_{ij} = \begin{cases} 
1, & \text{no previous motifs in } [X_{i,j}, \ldots, X_{i,j+w-1}] \\
0, & \text{otherwise}
\end{cases}$$

### 15.6 Motif Representation and Information Content

To begin our discussion, we will talk about uncertainty and probability. Uncertainty is related to our surprise at an event. The event “the sun will rise tomorrow” is not very surprising, so it’s uncertainty is quite low. However, the event “the sun will not rise tomorrow” is very surprising and it has a high uncertainty. In general, we can model uncertainty by $-\log p$.

The Shannon entropy is a measure of average uncertainty weighted by the probability of an event occurring. It is a measure of randomness. The logarithm is taken base two. The Shannon entropy is given by the equation:

$$H(X) = -\sum p_i \log_2 p_i$$

Entropy is maximum at maximum randomness. For example, if a biased coin is tossed, then the entropy is maximal when the coin is fair, that is the toss is most random.

Information is a decrease in uncertainty and is measured by the difference in entropy once you receive knowledge about a certain event. We can model a motif by how much uncertainty we lose after applying Gibbs Sampling or EM. In the following figure, the height of each stack represents the number of bits of uncertainty that each position is reduced by. Higher stacks correspond to much certainty about the base at the position of a motif while lower stacks correspond to a higher degree of uncertainty. Specifically, the height of a letter is 2 minus the entropy of that position. The entropy will be high if all bases are weighted equally and low if one base always occurs in one position. Thus, tall stacks correspond to positions where there is a lot of certainty. The height of a letter is proportional to the frequency of the base at that position.

There is a distance metric on probability distributions known as the Kullback-Leibler distance. This allows us to compare the divergence of the motif distribution to some true distribution. The K-L distance is given by

$$D_{KL}(P_{\text{motif}} | P_{\text{background}}) = \sum_{A,T,G,C} P_{\text{motif}}(i) \log \frac{P_{\text{motif}}(i)}{P_{\text{background}}(i)}$$

In Plasmodium, there is a lower G-C content. If we assume a G-C content of 20%, then we get the following representation for the above motif. C and G bases are much more unusual, so their prevalence is highly unusual. Note that in this representation, we used the K-L distance, so that it is possible for the stack to be higher than 2.
Figure 15.8: Entropy is maximized when the coin is fair, when the toss is most random.

Figure 15.9: The height of each stack represents the number of bits of uncertainty that each position is reduced by.

Figure 15.10: lexA binding site assuming low G-C content and using K-L distance.

15.7 Current Research Directions
15.8 Further Reading
15.9 Tools and Techniques
15.10 What Have We Learned?
Bibliography
CHAPTER SIXTEEN

REGULATORY GENOMICS

Lecturer: Pouya Kheradpour
Scribe: Maria Frendberg

List of figures
16.1 Introduction

Every cell has the same DNA, but they all have different expression patterns due to the temporal and spatial regulation of genes. Regulatory genomics explains these complex gene expression patterns. The regulators we will be discussing are:

- **Transcription Factor (TF)** - Regulates transcription of DNA to mRNA. TFs are proteins which bind to DNA before transcription and either increase or decrease transcription. We can determine the specificity of a TF through experimental methods using protein or antibodies. We can find the genes by their similarity to know TFs.

- **Micro RNA (miRNA)** - Regulates translation of mRNA to Proteins. miRNAs are RNA molecules which bind to mRNA after transcription and can reduce translation. We can determine the specificity of an miRNA through experimental methods, such as cloning, or computational methods, using conservation and structure.

16.1.1 History of the Field

16.1.2 Open Problems

Both TFs and miRNAs are regulators and we can find them through both experimental and computational methods. We will discuss some of these computational methods, specifically the use of evolutionary signatures. These regulators bind to specific patterns, called motifs. We can predict the motifs to which a regulator will bind using both experimental and computational methods. We will be discussing identification of miRNAs through evolutionary and structural signatures and the identification of both TFs and miRNAs through de novo comparative discovery, which theoretically can find all motifs. Given a motif, it is difficult to find the regulator which binds to it.

A target is a place where a factor binds. There are many sequence motifs, however many will not bind; only a subset will be targets. Targets for a specified regulator can be determined using experimental methods. In Lecture 11, methods for finding a motif given a target were discussed. We will also discuss finding targets given a motif.

![Figure 16.1: Challenges in Regulatory Genomics](image-url)
16.2  *De Novo* Motif Discovery

16.2.1 TF Motif Discovery

Transcription Factors influence the expression of target genes as either activators or repressors by binding to the DNA near genes. This binding is guided by TF sequence specificity. The closer the DNA is to the base preference, the more likely it is that the factor will bind. These motifs can be found both computationally and experimentally. There are three main approaches for discovering these motifs.

- **Co-Regulation** - In Lecture 11, we discussed a co-regulation type of discovery of motifs by finding sequences which are likely to have the motif bound. We can then use enumerative approaches or alignment methods to find these motifs in the upstream regions. We can apply similar techniques to experimental data where you know where motif is bound.

- **Factor Centric** - There are also factor centric methods for discovering motifs. These are mostly experimental methods which require a protein or antibody. Examples include SELEX, DIP-Chip, and PBM's. All of these methods are in vitro.

- **Evolutionary** - Instead of focusing on only one factor, evolutionary methods focus on all factors. We can begin by looking at a single factor and determining which properties we can exploit. There are certain sequences which are preferentially conserved (conservation islands). However, these are not always motifs and instead can be due to chance or non-motif conservation. We can then look at many regions, find more conserved motifs, and determine which ones are more conserved overall. By testing conservation in many regions across many genomes, we increase the power. These motifs have certain evolutionary signatures that help us to identify them: motifs are more conserved in intergenic regions than in coding regions, motifs are more likely to be upstream from a gene than downstream. This is a method for taking a known motif and testing if it is conserved.

We now want to find everything that is more conserved than expected. This can be done using a hill climbing approach. We begin by enumerating the motif seeds, which are typically in 3-gap-3 form. Then, each of these seeds is scored and ranked using a conservation ratio corrected for composition and small counts. These seeds are then expanded to fill unspecified bases around the seed using hill climbing. Through these methods, it is possible to arrive at the same, or very similar seeds in different manners. Thus, our final step consists of clustering the seeds using sequence similarity to remove redundancy.

A final method that we can use is recording the frequency with which one sequence is replaced by another in evolution. This produces clusters of k-mers that correspond to a single motif.

16.2.2 Validating Discovered Motifs

There are many ways that we can validate discovered motifs. Firstly, we expect them to match real motifs, which does happen significantly more often than with random motifs. However, this is not a perfect agreement, possibly due to the fact that many known motifs are not conserved and that known motifs are biased and may have missed real motifs. Positional bias. Biased towards TSS.

Motifs also have functional enrichments. If a specific TF is expressed in a tissue, then we expect the upstream region will have that factor’s motif. This also reveals modules of cooperating motifs. We also see that most motifs are avoided in ubiquitously expressed genes, so that they are not randomly turned on and off.

16.2.3 Summary

There are disadvantages to all of these approaches. Both TF and region-centric approaches are not comprehensive and are biased. TF centric approaches require a transcription factor or antibody, take lots of time and money, and also have computational challenges. De novo discovery using conservation is unbiased, but it can’t match motifs to factors and requires multiple genomes.
16.3 Predicting Regular Targets

16.3.1 Motif Instance Identification

Once potential motifs are discovered, the next step is to discover which motif matches are real. This can be done by both experimental and computational methods.

- **Experimental** - Instances can be identified experimentally using ChIP-Chip and ChIP-Deq methods. Both of these are in vivo methods. This is done by cross linking cells. DNA is first broken into sections. Then the protein and its antibody or tagged protein is added, which binds to various sequences. These bound sequences are now pulled out and cross linking is reversed. This allows us to determine where in the genome the factor was bound. This has a high false positive rate because there are many instances where a factor binds, but is not functional. This is a very popular experimental methods, but it is limited by the availability of antibodies, which are difficult to get for many factors.

- **Computational** - Computation approaches. There are also many computational approaches to identify instances. Single genome approaches use motif clustering. They look for many matches to increase power and are able to find regulatory regions (CRMs). However, they miss instances of motifs that occur alone and require a set of specific factors that act together. Multi-genome approaches, known as phylogenetic footprinting, face many challenges. They begin by aligning many sequences, but even in functional motifs, sequences can move, mutate, or be missing. The approach taken by Kheradpour handles this by not requiring perfect conservation (by using a branch length score) and by not requiring an exact alignment (by searching within a window).

Branch Length Scores (BLS) are computed by taking a motif match and searching for it in other species. Then, the smallest subtree containing all species with a motif match is found. The percentage of total tree is the BLS. Calculating the BLS in this way allows for mutations permitted by motif degeneracy, misalignment and movement within a window, and missing motifs in dense species trees.

This BLS is then translated into a confidence score. This enables us to evaluate the likelihood of a given score and to account for differences in motif composition and length. We calculate this confidence score by counting all motif instances and control motifs at each BLS. We then want to see which fraction of the motif instances seem to be real. The confidence score is then signal/(signal+noise). The control motifs used in this calculation are produced by producing 100 shuffles of the original motif, and filtering the results by requiring that they match the genome with +/- 20% of the original motif. These are then sorted based on their similarity to known motifs and clustered. At most one motif is taken from each cluster, in increasing order of similarity, to produce our control motifs.

16.3.2 Validating Targets

Similar to motif discovery, we can validate targets by seeing where they fall in the genome. Confidence selects for TF motif instances in promoters and miRNA motifs in 3’ UTRs, which is what we expect. TFs can occur on either strand, whereas miRNA must fall on only one strand. Thus, although there is no preference for TFs, miRNA are found preferentially on the plus strand.

Another method of validating targets is by computing enrichments. This requires having a background and foreground set of regions. These could be a promoter of co-regulated genes vs all genes or regions bound by a factor vs other intergenic regions. Enrichment is computed by taking the fraction of motif instances inside the foreground vs the fraction of bases in the foreground. Composition and conservation level are corrected for with control motifs. These fractions can be made more conservative using a binomial confidence interval.

Targets can then be validated by comparing to experimental instances found using ChIP-Seq. This shows the conserved CTCF motif instances are highly enriched in ChIP-Seq sites. Increasing confidence also increases enrichment. Using this, many motif instances are verified. ChIP-Seq does not always find functional motifs, so these results can further be verified by comparing to conserved bound regions. This finds that enrichment in intersections is dramatically higher. This shows where factors are binding that have an effect worthwhile conserving in evolution. These two approaches are complementary and are even more effective when used together.
16.4 MicroRNA Genes and Targets

16.4.1 MiRNA Gene Discovery

MiRNAs are post-transcriptional regulators that bind to miRNAs to silence a gene. They are an extremely important regulator in development. These are formed when a miRNA gene is transcribed from the genome. The resulting strand forms a hairpin at some point. This is processed, trimmed and exported to the cytoplasm. Then, another protein trims the hairpin and one half is incorporated into a RISK complex. By doing this, it is able to tell the RISK complex where to bind, which determines which gene is turned off. The second strand is usually discarded. It is a computational problem to determine which strand is which. The computational problem here is how to find the genes which correspond to these miRNAs.

The first problem is finding hairpins. Simply folding the genome produces approximately 760,000 hairpins, but there are only 60 to 200 true miRNAs. Thus we need methods to help improve specificity. Structural features, including folding energy, loops (number, symmetry), hairpin length and symmetry, substructures and pairings, can be considered, however, this only increases specificity by a factor of 40. Thus structure alone cannot predict miRNAs. Evolutionary signatures can also be considered. MiRNA show characteristic conservation properties. Hairpins consist of a loop, two arms and flanking regions. In most RNA, the loop is the most well conserved due to the fact that it is used in binding. In miRNA, however, the arms are more conserved because they determine where the RISK complex will bind. This increases specificity by a factor of 300. Both these structural features and conservation properties can be combined to better predict potential miRNAs.

These features are combined using machine learning, specifically random forests. This produces many weak classifiers (decision trees) on subsets of positives and negatives. Each tree then votes on the final classification of a given miRNA. Using this technique allows us to reach the desired sensitivity (increased by 4,500 fold).

16.4.2 Validating Discovered MiRNAs

Discovered miRNAs can be validated by comparing to known miRNAs. An example given in class shows that 81% of discovered miRNAs were already known to exist, which shows that these methods perform well. The putative miRNAs have yet to be tested, however this can be difficult to do as testing is done by cloning.

Region specificity is another method for validating miRNAs. In the background, hairpins are fairly evenly distributed between introns, exons, intergenic regions, and repeats and transposons. Increasing confidence in predictions causes almost all miRNAs to fall in introns and intergenic regions, as expected. These predictions also match sequencing reads.

This also produced some genomic properties typical of miRNAs. They have a preference for transcribed strand. This allows them to piggyback in intron of real gene, and thus not require a separate transcription. They also clustering with known and predicted miRNAs. This indicates that they are in the same family and have a common origin.

16.4.3 MiRNA’s 5’ End Identification

The first seven bases determine where an miRNA binds, thus it is important to know exactly where clevage occurs. If this clevage point is wrong by even two bases, the miRNA will be predicted to bind to a completely different gene. These clevage points can be discovered computationally by searching for highly conserved 7-mers which could be targets. These 7-mers also correlate to a lack of anti-targets in ubiquitously expressed genes. Using these features, structural features and conservational features, it is possible to take a machine learning approach (SVMs) to predict clevage site. Some miRNAs have no single high scoring position, and these also show imprecise processing in the cell. If the star sequence is highly scored, then it tends to be more expressed in the cell also.

16.4.4 Functional Motifs in Coding Regions

Each motif type has distinct signatures. DNA is strand symmetric, RNA is strand-specific and frame-invariant, and Protein is strand-specific and frame-biased. This frame-invariance can be used as a signature.
Each frame can then be evaluated separately. Motifs due to di-codon usage biases are conserved in only one frame offset while motifs due to RNA-level regulation are conserved in all three frame offsets. This allows the ability to distinguish overlapping pressures.

16.5 Current Research Directions

16.6 Further Reading

16.7 Tools and Techniques

16.8 What Have We Learned?

Bibliography
TODO: missing @scribe: who was this scribed by?

List of figures

Chromatin Immunoprecipitation ??
Burows-Wheeler Transform ??
Chromatin State Definition and Functional Interpretation ??
17.1 Introduction

Within the human body there are approximately 210 different cell types all of which share the same genomic sequence. Cells not only develop into distinct types from this same sequence, they also maintain the same cell type over time and across divisions (unless perturbed). This information about the cell type and the state of the cell is called epigenomic information. Most of this information is contained in the way DNA is packaged. In order to fit two meters of DNA into a 5-20-diameter cell nucleus and arrange the DNA for easy access to transcriptional machinery, DNA is packaged into chromatin. Nucleosome is the main player of this packaging; it is composed of eight histone proteins (named histone H2A, H2B, H3, H4) and occasionally a linker histone H1 or H5. Each unique histone appears twice in the octamer) with DNA wrapped around the outside of this ball of histones in two revolutions. Each nucleosome consumes approximately 150-200 bp of DNA. While the structure and importance of higher-level packaging of nucleosomes is less known, the lower-level arrangement and modification of nucleosomes is very important to transcriptional regulation and the development of different cell types. Histone proteins H3 and H4 are the most highly conserved proteins in the eukaryotic domain of life. If DNA contains the blueprints of life, nucleosomes contain the blueprints of multi-cellular life.

Nucleosomes have several features that contain epigenetic information. First, their positions on the DNA are important. Nucleosomes are often bound to the promoters of inactive genes. To initiate transcription of a gene, transcription factors (TFs) and the General Factors have to bind to its promoter. Therefore, when a gene becomes active, the nucleosomes located on its promoter are often pushed aside or removed. The promoter will remain exposed until further modifications are made. Hence, nucleosome positioning on the DNA is stable, yet mutable. This property of stability and mutability is a prerequisite for any form of epigenetic information because cells need to maintain the identity of a particular cell type, yet still be able to change their epigeneti state to respond to environmental circumstances

Second, nucleosomes contain tails of amino acid residues protruding from the ends of their histones. These tails can undergo modification such as methylation, acetylation and phosphorylation. For instance, the fourth residue from the N-terminus of histone H3, lysine, is often methylated at the promoters of active genes. Modifications of this nature are so common that a shorthand notation has been developed. The above modification would be described as H3K4me3 if it were methylated three times. H3 = histone H3, K4 = lysine, 4th residue from end, me3 = 3 methyl groups added.

Lastly, epigenetic information can also be stored by DNA methylation. DNA is often methylated at CpG islands. Methylated cytosine is more likely to mutate into thiamine, which is why transition matrices weight C → T transitions so heavily. Adenosine, in rare cases, may also be methylated.

17.2 Technologies for measurement of epigenetic signals

Given the importance of epigenomic information in biology, great efforts have been made to study these signals on DNA. One common method for epigenomic mark measurement is chromatin immunoprecipitation (ChIP). The procedures of ChIP are described as follows:

1. Cells are exposed to a cross-linking agent such as formaldehyde, which causes covalent bonds to form between DNA and its bound proteins (e.g. histones with specific modifications);

2. Genomic DNA is isolated from the cell nucleus;

3. Isolated DNA is sheared by sonication or enzymes;

4. Antibodies against a specific epigenetic mark are then added to pull out its associated DNA. These antibodies are generated by exposing proteins of interest to mammals (e.g. goats or rats). The resulting immune response will cause the production of specific antibodies.

5. The cross-linking between the protein and DNA is reversed and the DNA fragments specific to the epigenetic marks are purified.

To identify these DNA fragments, one can hybridize them to known DNA segments on an array or gene chip and visualize them with fluorescent marks (called ChIP-chip). Alternatively, one can do massive parallel
next-generation sequencing of these fragments (called ChIP-seq). Each sequence tag is 30 base pairs long. These tags are mapped to unique positions in the reference genome of 3 billion bases. The number of reads depending on sequencing depth, but typically there are on the order of 10 million mapped reads for each ChIP-seq experiment. ChIP-seq is preferred over ChIP-chip nowadays because it has wider dynamic range of detection and can avoid problems such as cross-hybridization in ChIP-chip. Given the vast amount of reads and the size of human genome, one common problem encountered in all high-throughput sequencing is the algorithmic efficiency of assigning reads to the best matching location. There are two major approaches in matching short reads to the genome. The first one is a hashtable-based approach such as MAQ, which basically trying to put all the reads in a hashtable and use it to search for matches in the genome. The second approach, Burrows-Wheeler transformation, is preferred because it is much faster than MAQ. The details of Burrows-Wheeler transformation are described as follows and illustrated in ??:

1. For a given reference genome, add a special character (e.g. $) and then generate all the orations.
2. Sort strings lexicographically, i.e. in alphabetical order based on the first column of each string, with special characters sorted first.
3. The last column of the sorted list of strings contains the transformed genome. An example is given in ??a.

17.2.1 Key Properties of Such Transformation:

1. All occurrences of the same suffix are effectively next to each other rather than scattered throughout the genome, as illustrated in ??c.
2. The $i^{th}$ occurrence of a character in the first column corresponds to the $i^{th}$ occurrence in the last column, as illustrated in ??b.

After reads are aligned, signal tracks can be computed. This data can be ordered into a long histogram spanning the length of the genome and indicating the number of reads (or degree of fluorescence in the case of ChIP-chip) found at each position in the genome. More reads (or fluorescence) means the epigenomic marker of interest is most often present at this particular location. For histone modifications, peak/internal calling methods are based on univariate Hidden Markov Model, or scan statistics (count the reads within
6.047/6.878 Lecture 17: Epigenomics/Chromatin Sites

Figure 17.2: a. The Burrows-Wheeler matrix and transformation for ‘acaacg’. b. Repeatedly apply the last first (LF) mapping to recover the original text (in red on the top line) from the Burrows-Wheeler transform (in black in the rightmost column). c. Steps taken by to identify the range of rows, and thus the set of reference suffixes, prefixed by ‘aac’.?

bins of certain size and apply statistical analysis). Problems tend to arise with broader domains due to the ambiguity of calling them one large peak or multiple smaller peaks.

There are many ways to analyze epigenomic marks. For instance, one can aggregate signal (such as H3K4me3) on known feature types (e.g. promoters of genes with high or low expression levels). Additionally, one can incorporate machine learning methods to derive epigenomic features that are predictive of different types of genomics elements such as promoters, enhancers or large intergenic non-coding RNAs.

17.3 Annotating the Genome Using Chromatin Signatures

17.3.1 Formulation of the Problem:

Epigenetic factors play an important role biologically. They can recruit protein elements or conversely to inhibit DNA access. Over 100 distinct histone modifications have been described. Different combinations of these chromatin modifications, when taken together, can help determine how a region of DNA is interpreted by the cell (i.e. as a transcription factor binding domain, a splice site, an enhancer region, an actively expressed gene, a repressed gene, or a non functional region). Stated another way, DNA can take on a series of (hidden) states (coding, noncoding, etc). Each of these states emits a specific combination of epigenetic modifications (H3K4me3, H3K36me3, etc) that the cell recognizes. We want to be able to predict these hidden, biologically relevant states from observed epigenetic modifications.

17.3.2 The Solution:

Use a multivariate Hidden Markov Model (HMM) for the de novo identification of chromatin states, or biologically meaningful and spatially coherent combinations of chromatin marks, in a given cell population (e.g. CD4+ T-cells). As stated above, chromatin states are considered our hidden states. States can switch to other states with certain transition probabilities, and each state emits a certain combination of epigenetic markers. This model can capture both the functional ordering of different states (e.g from promoter to transcribed regions) and the spreading of certain chromatin domains across the genomes. Since we do not
know what biological chromatin states exist (and how many there are), Baum-Welch training will be used. There are a few differences between this model and the HMM that we have studied in the past:

**Emission of a Vector**

In HMM models from previous lectures, each state would emit either a single nucleotide or a single string of nucleotides at a time. In this HMM model, each state emits a combination of epigenetic marks with a specific probability. Each combination can be represented as an n-dimensional vector where n is the number of epigenetic modification being analyzed. For example, assuming you have four possible epigenetic modifications: H3K4me3, H2BK5ac, Methyl-C, and Methyl-A, a sequence containing H3K4me3 and Methyl-C could be presented as the vector (1, 0, 1, 0). This data binarization can generate biologically interpretable models that can be robustly learned.

However, a present call for each epigenetic mark is not as exact as discrete nucleotides A,G,C or T. Here, let $C_{ij}$ be the number of reads for mark i detected by ChIP-seq, mapping to bin j. Let $\lambda_i$ be the average number of reads mapping to a bin for mark i. The input for feature i becomes 1 if $P(X > C_{ij}) < 10^{-4}$ where X is a Poisson random variable with mean $\lambda_i$. In other words, the read enrichment for a specific mark has to be significantly greater than a random process of putting reads into bins.

Additionally, for M input marks each state k has a vector of $(p_{k1}, ..., p_{kM})$ of parameters for independent Bernoulli random variables to determine the emission probability for an observed combination of marks.

**State Lengths**

Since the emissions are no longer nucleotide sequences, but epigenetic marks, it becomes necessary to set a specific length for each state so that the presence or absence of an epigenetic mark can be measured in this region. For the HMM discussed in class a length of 200 nucleotides was established.

17.3.3 Choosing the Number of states to model

As with most machine learning algorithms, increasing the number of categories (i.e. hidden states) during training will increase the variability captured by a model. However, part of this variability is due to limited sampling of the true population in our training set. Extra categories can only reduce the variability in a particular sample, but not the true population. This is often called over-fitting of the data. Bayesian Information Criterion (BIC) is a common technique for optimizing the number of states permitted in a model given finite training data. Using BIC, one can visualize the increasing power of a model as a function of the number of states. Generally, one will choose a value for k (the number of states) such that the addition of more states has relatively little benefit in terms of predictive power gain. However, even with BIC, the resulting model is likely to have more states than an ideal model with most useful states for biological interpretation. The reason is that the human genome is so big that statistically significant differences are easy to find, yet many of these differences are not biologically significant.

To solve this problem, a model with 79 states was initially generated based on the best BIC score after being trained on a population of CD4+ T cells using Baum-Welch algorithm. Of note, in Baum-Welch training, different initializations can turn into very different resulting models. In other words, different initialization can confound the result when doing comparisons among models. Hence, nested initialization rather than randomized initialization was used in generating a model. A single state was greedily removed from the 79-state model. When removing a state the emission probabilities would be removed entirely, and any state that transitioned to it would have that transition probability uniformly redistributed to all the remaining states. The remaining 78 states were used to initialize another round of Baum-Welch training. The correlation between this model and the initial 78 state model was calculated and this process was repeated for all 1-78 state models. This procedure was repeated three times and the model with the maximum correlation was saved. The number of states for a model to analyze can then be selected by choosing the model trained from such nested initialization with the smallest number of states that sufficiently captures all states offering distinct biological interpretations. The resulting final model had 51 states and it was further verified by showing the independence of individual modifications in each class, demonstrating that the majority of information contained in the epigenetic modifications was captured.
17.3.4 Results

This multivariate HMM model resulted in a set of 51 biologically relevant chromatin states. However, there were no one-to-one relationship between each state and known classes of genomic elements (e.g. introns, exons, promoters, enhancers, etc) Instead, multiple chromatin states were often associated with one genomic element. Each chromatin state encoded specific biological relevant information about its associated genomic element. For instance, three different chromatin states were associated with transcription start site (TSS), but one was associated with TSS of highly expressed genes, while the other two were associated with TSS of medium and lowly expressed genes respectively. Such use of epigenetic markers greatly improved genome annotation, particularly when combined with evolutionary signals discussed in previous lectures. The 51 chromatin states can be divided in five large groups. The properties of these groups are described as follows and further illustrated in ??:

1. **Promoter-Associated States (1-11):**
   
   These chromatin states all had high enrichment for promoter regions. 40-89% of each state was within 2 kb of a RefSeq TSS. These states all had a high frequency of H3K4me3, significant enrichments for DNase I hypersensitive sites, CpG islands, evolutionarily conserved motifs and bound transcription factors. These states differed in their functional enrichment based on Gene Ontology (GO). For instance, genes associated with T cell activation were enriched in state 8 while genes associated with embryonic development were enriched in state 4. Additionally, among these promoter states there were distinct positional enrichments. States 1 -3 peaked both upstream and downstream of TSS; states 4-7 were concentrated right over TSS whereas states 8-11 peaked between 400 bp and 1200 bp downstream of TSS.

2. **Transcription-Associated States (12-28):**

   These states were not predominantly associated with a single mark but rather they were defined by a combination of several marks. Some of these states were associated with spliced exons, transcription start sites or end sites. Of interest, state 28, which was characterized by high frequency for H3K9me3, H4K20me3, and H3K36me3, showed a high enrichment in zinc-finger genes. This specific combination of marks was previously reported as marking regions of KAP1 binding, a zinc-finger specific co-repressor.

3. **Active Intergenic States (29-39):**

   These states were associated with several classes of candidate enhancer regions and insulator regions. These regions were usually away from promoters and were outside of transcribed genes. Interestingly, several active intergenic states showed a significant enrichment for disease SNPs, or single nucleotide polymorphism in genome-wide association study (GWAS). For instance, a SNP (rs12619285) associated with plasma eosinophil count levels in inflammatory diseases was found to be located in the chromatin state 33, which was enriched for GWAS hits. In contrast, the surrounding region of this SNP was assigned to other chromatin states with no significant GWAS association. This can shed light on the possible functional significance of disease SNPs based on its distinct chromatin states.

4. **Large-Scale Repressed States (40-45):**

   These states marked large-scale repressed and heterochromatic regions, representing 64% of the genome. H3K27me3 and H3K9me3 were two most frequently detected marks in this group.

5. **Repetitive States (46-51):**

   These states showed strong and distinct enrichments for specific repetitive elements. For instance, state 46 had a strong sequence signature of low-complexity repeats such as (CA)n, (TG)n, and (CATG)n.

   a. Chromatin mark combinations associated with each state. Each row shows the specific combination of marks associated with each chromatin state and the frequencies between 0 and 1 with which they occur in color scale. These correspond to the emission probability parameters of the HMM learned across the genome during model training.
b. Genomic and functional enrichments of chromatin states, including fold enrichment in different part of the genome (e.g. transcribed regions, TSS, RefSeq 5' or 3'end of the gene etc), in addition to fold enrichment for evolutionarily conserved elements, DNaseI hypersensitive sites, CpG islands, etc. All enrichments are based on the posterior probability assignments.

c. Brief description of biological state function and interpretation (chr, chromatin; enh, enhancer).

The predictive power of chromatin states for discovery of functional elements consistently outperformed predictions based on individual marks. Such unsupervised model using epigenomic mark combination and spatial genomic information performed as well as many supervised models in genome annotation. It was shown that this HMM model based on chromatin states was able to reveal previously unannotated promoters and transcribed regions that were supported by independent experimental evidence. Hence, such genome-wide annotation based on chromatin states can help better interpret biological data and potentially discover new classes of functional elements in the genome.

17.3.5 Encode

All of the above work was done in a single cell type (CD4+ T cells). Since epigenomic markers vary over time, across cell types, and environmental circumstances, it is important to consider the dynamics of the
chromatin states across different cell types and experimental conditions. The ENCODE project \footnote{ENCODE project} has measured 11 different chromatin marks in eight different cell types. Each ChIP-seq experiment in the project produced 107 reads, so the total quantity of data is immense and requires computational analysis. A similar multivariate HMM model was developed for each of these cell types resulting in 15 states. Each cell type was analyzed for class enrichment. It was shown that some chromatin states, such as those encoding active promoters were highly stable across all cell types. Other states, such as those encoding strong enhancers, were highly enriched in a cell-type specific manner, suggesting their roles in tissue specific gene expression. Finally, it was shown that there was significant correlation between the epigenetic marks on enhancers and the epigenetic marks on the genes they regulate, even though these can be thousands of base pairs away. Such chromatin state model has proven useful in matching enhancers to their respective genes, a problem that has been largely unsolved in modern biology.

17.4 Current Research Directions

17.5 Further Reading

17.6 Tools and Techniques

17.7 What Have We Learned?

Bibliography


CHAPTER

EIGHTEEN

REGULATORY NETWORKS: INERENCE, ANALYSIS, APPLICATION

Guest Lecture by
Sushmita Roy
Scribed by Ben Holmes

List of figures

TODO: missing @scribe: INSERT LIST OF FIGURES, changes references to slide numbers to actually include important images from the slides
18.1 Introduction

Living systems are divided into layers. Layers have parts.

1. Epigomics layer is composed of nucleosomes and nucleosome free regions. It organises DNA.¹

2. DNA layer is composed of coding and non coding junk DNA. It stores the code from which RNAs and eventually proteins are built.

3. RNA layer is composed of transcribed RNAs that code proteins and noncoding RNAs, the transcriptome. It has regulatory functions and protein manufacturing functions.

4. Protein layer is composed of protein parts, TFs, signalling proteins, metabolic enzymes, etc.

Genomics has successfully characterized many of these layers. Now the goal is to put them into context and combine them. This is the job of networks.

18.1.1 Introducing Biological Networks

Regulatory Net – Edges correspond to regulatory interactions and nodes are regulators and associated targets. This is a directed graph describing the set of regulatory interactions in an organism.

Metabolic Net – Edges correspond to regulatory reactions and nodes are enzymes. An undirected weighted graph connecting metabolic processes. There is some flexibility in the choice of a specific representation. In the representation visualized, edges represent nodes sharing metabolic products.

Signaling Net – Nodes are signaling receptors on cellular surfaces transmitting and receiving biological signals corresponding to edges in a directed graph.

Protein Net – Nodes are proteins, edges correspond to physical interactions between proteins. Undirected graph.

Co-Expression Net – Very general: nodes are genes and edges describe co-expression. A little bit different from the rest as it represents functional rather than physical interaction networks. Important powerful tool in computational analysis of biological data. Undirected graph.

18.1.2 Interactions Between Biological Networks

Individual biological networks (layers) can themselves be considered nodes in a larger network representing the entire biological system. We can for example have a signaling network sensing the environment governing the expression of transcription factors. TFs govern the expression of proteins, proteins can play roles as enzymes in metabolic pathways,... Everything interacts but today’s class will focus on regulatory networks.²

18.1.3 Why Regulatory Networks?

Why study regulatory networks? Because many things can go wrong with them

- This is the layer where you have the maximum amount of control

¹More in the epigenetics lecture.
²Dr. Roy is happy to talk to anyone about any other networks.
• Many diseases are caused by rewirings of regulatory networks.
• Controls context specific expression in development.
• Can be used in systems biology to predict development, cell state, system state etc!
• Encapsulates much of the evolutionary difference between organisms that are not so different genetically.

18.1.4 Challenges in Regulatory Networks

Regulatory network description presents several distinct challenges.

**Element Identification** What are the elements of a network? Elements constituting regulatory networks were identified last lecture. These include upstream motifs and their associated factors.

**Network Structure Analysis** How are the elements of a network connected? Given a network, structure analysis consists of examination and characterization of important properties. It can be done biological networks but is not restricted to them.

**Network Inference** How do regulators interact and turn on genes? This is the task of identifying gene edges and characterizing their actions.

**Network Applications** What can we do with networks once we have them? Applications include predicting function of regulating genes and predicting expression levels of regulated genes.

18.2 Structural Properties of Networks

Much of the early work on networks was done by scientists outside of biology. Physicists looked at internet and social networks and described their properties. Biologists observed that the same properties cropped up in biological networks and the field of biological networks was born.

18.2.1 Fundamental Properties of Networks

Four main properties describe networks once nodes have been established.

**Degree Distribution** A probability distribution for every node. Many networks are scale free and cannot be said to have average nodes. Scale free networks have hubs with high connectivity.

**Modularity** Clusters of nodes. Network modularity is a useful tool for distinguishing functional subgraphs of a network and suggesting function for unknown nodes.

**Hierarchical Nature** Decomposition into levels. Networks can be split into hierarchical levels differentiated by their populations, tendency to regulate directly, and essentiality.

**Network Motifs** Frequently occurring subgraphs in a network can be associated with key biological functions.

18.2.2 Scale Free Distributions

Scale free networks are self similar and follow a power law degree distribution. They have hubs but no average node. You can travel quickly through a network by hubs but if you knock out a hub, you lose significant portions of the network. Some regulatory networks are scale free. In the figure we see one that is: the red line represents in degree of *S. cerevisiae* and the white dotted line is a random network. The in degree of *S. cerevisiae* is much closer to scale free than the random network.
Hierarchical Nature of Networks

Four levels:

1. Influential, master regulating nodes on top. These are hubs that each indirectly control many targets. Controlled by signalling networks.

2. Bottle neck regulators. Nodes in the middle are important because they have a maximal number of direct targets.

3. Regulators at the bottom tend to have fewer targets but nonetheless they are often biologically essential!

4. Targets.

18.2.3 Modularity of Networks

Network modularity allows decomposition of networks into distinct parts whose subparts may be associated with particular functions. With anumber of algorithms, we can partition the network into clusters we will associate with submodules.

Markov Clustering Algorithm By modeling a random walk as a markov random process you can locate densely connected components of a graph.

Girvan-Newman Algorithm Uses the number of shortest paths going through a node to compute essentiality of an edge. Attempts to retain interaction highways while splitting weak edges in orde to clarify clusters.

18.2.4 Network Motifs

Network motifs are subgraphs that occur significantly more than random. Some will have interesting functional properties and are presumably of biological interest.

Figure TODO: missing @scribe: insert figure from corresponding lecture slide shows regulatory motifs from the yeast regulatory network. Feedback loops allow control of regulator levels and feedforward loops allow accelera- tion of response times among other things. ³

18.2.5 Models for Inteference

Most regulatory networks are unknown. even in model organisms, we often do not specifically understand functional elements of networks. How do nodes interact and why?

We will discuss models with which to describe networks as well as various methods with which to learn them.

18.3 How to Find Structure?

³Dr. Roy is happy to discuss other network motifs outside of class
18.3.1 Key Questions in Structure Inference

*slide 15*

**How to choose network models?** A number of models exist for representing networks, a key problem is choosing between them based on data and predicted dynamics.

**How to choose learning methods?** Two broad methods exist for learning networks. Unsupervised methods attempt to infer relationships for unlabeled datapoints and will be described in sections to come. Supervised methods take a subset of network edges known to be regulatory, and learn a classifier to predict new ones.4

**How to incorporate data?** A variety of data sources can be used to learn and build networks including Motifs, ChIP binding assays, and expression. Data sources are always expanding; expanding availability of data is at the heart of the current revolution in analyzing biological networks.

18.3.2 Abstract Mathematical Representations for Networks

*slide 16*

Think of a network as a function, a black box. Regulatory networks for example, take input expressions of regulators and spit out output expression of targets. Models differ in choosing the nature of functions and assigning meaning to nodes and edges.

**Boolean Network** This model discretizes node expression levels and interactions. Functions represented by edges are logic gates.

**Differential Equation Model** These models capture network dynamics. Expression rate changes are function of expression levels and rates of change of regulators. For these it can be very difficult to estimate parameters. Where do you find data for systems out of equilibrium?

**Probabilistic Graphical Model** These systems model networks as a joint probability distribution over random variables. Edges represent conditional dependencies. Probabilistic graphical models (PGMs) are focused on in the lecture.

**Probabilistic Graphical Models**

*slide 17*

Probabilistic graphical models (PGMs) are trainable and able to deal with noise and thus they are good tools for working with biological data.5 In PGMs, nodes can be transcription factors or genes and they are modeled by random variables. If you know the joint distribution over these random variables, you can build the network as a PGMs. Since this graph structure is a compact representation of the network, we can work with it easily and accomplish learning tasks. Examples of PGMs include:

**Bayesian Network** Directed graphical technique. Every node is either a parent or a child. Parents fully determine the state of children but their states may not be available to the experimenter. The network structure describes the full joint probability distribution of the network as a product of individual distributions for the nodes. By breaking up the network into local potentials, computational complexity is drastically reduced.

**Dynamic Bayesian Network** Directed graphical technique. Static bayesian networks do not allow cyclic dependencies but we can try to model them with bayesian networks allowing arbitrary dependencies between nodes at different time points. Thus cyclic dependencies are allowed as the network progresses through time and the network joint probability itself can be described as a joint over all times.

---

4 Supervised methods will not be addressed today.

5 These are Dr. Roys models of choice for dealing with biological nets.
Markov Random Field Undirected graphical technique. Models potentials in terms of cliques. Allows modeling of general graphs including cyclic ones with higher order than pairwise dependencies.

Factor Graph Undirected graphical technique. Factor graphs introduce factor nodes specifying interaction potentials along edges. Factor nodes can also be introduced to model higher order potentials than pairwise.

It is easiest to learn networks for Bayesian models. Markov random fields and factor graphs require determination of a tricky partition function. To encode network structure, it is only necessary to assign random variables to TFs and genes and then model the joint probability distribution.

Bayesian networks provide compact representations of JPD

The main strength of Bayesian networks comes from the simplicity of their decomposition into parents and children. Because the networks are directed, the full joint probability distribution decomposes into a product of conditional distributions, one for each node in the network.\(^6\)

Network Inference from Expression Data

Using expression data and prior knowledge, the goal of network inference is to produce a network graph. Graphs will be undirected or directed. Regulatory networks for example will often be directed while expression nets for example will be undirected.

18.4 Overview of the PGM Learning Task

We have to learn parameters from the data we have. Once we have a set of parameters, we have to use parametrizations to learn structure. We will focus on score based approaches to network building, defining a score to be optimized as a metric for network construction.

18.4.1 Parameter Learning for Bayesian Networks

Maximum Likelihood Chooses parameters to maximize the likelihood of the available data given the model.

In maximum likelihood, compute data likelihood as scores of each random variable given parents and note that scores can be optimized independently. Depending on the choice of a model, scores will be maximized in different manners. For gaussian distribution it is possible to simply compute parameters optimizing score. For more complicated model choices it may be necessary to do gradient descent.

Bayesian Parameter Estimation Treats \(\theta\) itself as a random variable and chooses the parameters maximizing the posterior probability. These methods require a fixed structure and seek to choose internal parameters maximizing score.

---

\(^6\)Bayesian networks are parametrized by \(\theta\) according to our specific choice of network model. With different choices of random variables, we will have different options for parametrizations, \(\theta\) and therefore different learning tasks:

- **Discrete** Random variables suggest simple \(\theta\) corresponding to parameter choices for a multinomial distribution.
- **Continuous** Random variables may be modelled with \(\theta\) corresponding to means and covariances of gaussians or other continuous distribution.
Structure Learning  

We can compute best guess parametrizations of structured networks. How do we find structures themselves? Structure learning proceeds by comparing likelihood of ML parametrizations across different graph structures and in order to seek those structures realizing optimal of ML score.

A Bayesian framework can incorporate prior probabilities over graph structures if given some reason to believe a-priori that some structures are more likely than others.

To perform search in structure learning, we will inevitably have to use a greedy approach because the space of structures is too large to enumerate. Such methods will proceed by an incremental search analogous to gradient descent optimization to find ML parametrizations.

In the illustration of the slide TODO: missing @scribe: insert slide drawing from slide 26 in lecture 11, one such search path is demonstrated. A set of graphs are considered and evaluated according to ML score. Since local optima can exist, it is good to seed graph searches from multiple starting points.

Besides being unable to capture cyclic dependencies as mentioned above, Bayesian networks have certain other limitations.

Indirect Links Since Bayesian networks simply look at statistical dependencies between nodes, it is easy for them to be tricked into putting edges where only indirect relations are in fact present.

Neglected Interactions Especially when structural scores are locally optimized, it is possible that significant biological interactions will be missed entirely. Coexpressed genes may not share proper regulators.

Slow Speed Bayesian methods so far discussed are too slow to work effectively whole-genome data.

Excluding Indirect Links

How to eliminate indirect links? Information theoretic approaches can be used to remove extraneous links by pruning network structures to remove redundant information. Two methods are described.

ARACNE For every triplet of edges, a mutual information score is computed and the ARACNE algorithm excludes edges with the least information subject to certain thresholds above which minimal edges are kept.

MRNET Maximizes dependence between regulators and targets while minimizing the amount of redundant information shared between regulators by stripping edges corresponding to regulators with low variance.

Alternately it is possible to simply look at regulatory motifs and eliminate regulation edges not predicted by common motifs.

18.4.2 Learning Regulatory Programs for Modules

How to fix omissions for coregulated genes? By learning parameters for regulatory models instead of individual genes, it is possible to exploit the tendency of coexpressed genes to be regulated similarly. Similar to the method of using regulatory motifs to prune redundant edges, by modeling modules at once, we reduce network edge counts while increasing data volume to work with.

With extensions, it is possible to model cyclic dependencies as well. Module networks allow clustering revisititation where genes are reassigned to clusters based on how well they are predicted by a regulatory program for a module.

Modules however cannot accomodate genes sharing module membership. divide and conquer for speeding up learning
How to speed up learning? Dr. Roy has developed a method to break the large learning problem into smaller tasks using a divide and conquer technique for undirected graphs. By starting with clusters it is possible to infer regulatory networks for individual clusters then cross edges, reassign genes, and iterate.

18.4.3 Conclusions in Network Inference

Regulatory networks are important but hard to construct in general. By exploiting modularity, it is often possible to find reliable structures for graphs and subgraphs. Many extensions are on the horizon for regulatory networks. These include inferring causal edges from expression correlations, learning how to share genes between clusters, and others.

18.5 Applications of Networks

Using linear regression and regression trees, we will try to predict expression from networks. Using collective classification and relaxation labeling, we will try to assign function to unknown network elements.

We would like to use networks to:

1. predict the expression of genes from regulators.

In expression prediction, the goal is to parametrize a relationship giving gene expression levels from regulator expression levels. It can be solved in various manners including regression and is related to the problem of finding functional networks.

2. predict functions for unknown genes.

18.5.1 Overview of Functional Models

One model for prediction is a conditional gaussian: a simple model trained by linear regression. A more complex prediction model is a regression tree trained by nonlinear regression.

Conditional Gaussian Models

Conditional gaussian models predict over a continuous space and are trained by a simple linear regression to maximize likelihood of data. They predict targets whose expression levels are means of gaussians over regulators.

Conditional gaussian learning takes a structured, directed net with targets and regulating transcription factors. You can estimate gaussian parameters, $\mu$, $\sigma$ from the the data by finding parameters maximizing likelihood - after a derivative, the ML approach reduces to solving a linear equation.
From a functional regulatory network derived from multiple data sources\(^8\), Dr. Roy trained a gaussian model for prediction using time course expression data and tested it on a hold-out testing set. In comparisons to predictions by a model trained from a random network, found out that the network predicted substantially better than random.

In the line plots TODO: missing @scribe: insert plots from slide 40, blue lines are predicted and black lines are true levels for a gene. Remaining lines are regulator expression levels.

The linear model used makes a strong assumption on linearity of interaction. This is probably not a very accurate assumption to make but it appears to work to some extent with the dataset tested.

**Regression Tree Models**

Regression tree models allow the modeler to use a multimodal distribution incorporating nonlinear dependencies between regulator and target gene expression. The final structure of a regression tree describes expression grammar in terms of a series of choices made at regression tree nodes. Because targets can share regulatory programs, notions of recurring motifs may be incorporated. Regression trees are rich models but tricky to learn. regression trees in predicting expression

In practice, prediction works its way down a regression tree given regulator expression levels. Upon reaching the leaf nodes of the regression tree, a prediction for gene expression is made.

The slide pictures activating regulation by a pair of regulators on gene.

TODO: missing @scribe: insert image from slide 42

### 18.5.2 Functional Prediction for Unannotated Nodes

Given a network with an incomplete set of labels, the goal of function annotation is to predict labels for unknown genes. We will use methods falling under the broad category of guilt by association. If we know nothing about a node but that its neighbors are involved in a function, assign that function to the unknown node.

Association can include any notion of network relatedness discussed above such as co-expression, protein-protein interactions and co-regulation. Many methods work, two will be discussed: collective classification and relaxation classification; both of which work for regulatory networks encoded as undirected graphs.

**Collective Classification**

View functional prediction as a classification problem: Given a node, what is its regulatory class?.

In order to use the graph structure in the prediction problem, we capture properties of the neighborhood of a gene in relational attribute. Since all points are connected in a network, data points are no longer independently distributed - the prediction problem becomes substantially harder than a standard classification problem.

Iterative classification is a simple method with which to solve the classification problem. Starting with an initial guess for unlabeled genes it infers labels iteratively, allowing changed labels to influence node label predictions in a manner similar to gibbs sampling\(^9\).

Relaxation labeling is another approach originally developed to trace terrorist networks. The model uses a suspicion score where nodes are labeled with a suspiciousness according to the suspiciousness of its neighbors.

---

\(^8\)data sources included chromatin, physical binding, expression, motif

\(^9\)see the previous lecture by Manolis describing motif discovery
The method is called relaxation labeling because it gradually settles on to a solution according to a learning parameter. It is another instance of iterative learning where genes are assigned probabilities of having a given function.

Regulatory Networks for Function Prediction

For pairs of nodes, compute a regulatory similarity – the interaction quantity – equal to the size of the intersection of their regulators divided by the size of their union. Having this interaction similarity in the form of an undirected graph over network targets, can use clusters derived from a network in final functional classification.

The model is successful in predicting invaginal disk and neural system development. The blue line shows the score of every gene predicting its participation in neural system development. TODO: missing @scribe: insert figure from slide 50

Co-expression and co-regulation can be used side by side to augment the set of genes known to participate in neural system development.

18.6 Conclusions

Using the methods (linear, nonlinear learning via iterative or global methods) and models (directed graphs, undirected graphs, multinomials, gaussians,...) described in this talk, it is possible to infer structure and connection strengths of biological networks.

Having computed and modeled these networks, it is possible to perform many biologically relevant tasks including predicting gene expression and annotating new functions mysterious genes. Many avenues remain open for exploration.

18.7 Questions?

contact Sushmita Roy: sroy@broadinstitute.org

18.8 Current Research Directions

18.9 Further Reading

18.10 Tools and Techniques

18.11 What Have We Learned?

Bibliography
INTRODUCTION TO STEADY STATE METABOLIC MODELING

Guest Lecture by James Galagan
Scribed by Jake Shapiro, Andrew Shum, and Ashutosh Singhal (1910)
Molly Ford Dacus and Anand Oza (1909)
Christopher Garay (1908)

List of figures

Process leading to and including the citric acid cycle  ??
Adding constraints to the extreme pathways  ??
Maximizing two functions with linear programming  ??
In Silico Detection Analysis  ??
Constraining Solution Space to Create New Optimal Flux  ??
Flux Balance Analysis Model  ??
Flow in predicting state of a metabolic system under varying drug treatments  ??
Constraining of cone shape.  ??
Nutrient source prediction experiment.  ??

195
19.1 Introduction

Metabolic modeling allows us to use mathematical models to represent complex biological systems. This lecture discusses the role of modeling the steady state of biological systems in understanding the metabolic capabilities of interesting organisms. We also briefly discuss how well steady state models are able to replicate in-vitro experiments.

19.1.1 What is Metabolism?

According to Matthews and van Holde, metabolism is the totality of all chemical reactions that occur in living matter. This includes catabolic reactions, which are reactions that lead to the breakdown of molecules into smaller components, and anabolic reactions, which are responsible for the creation of more complex molecules (e.g., proteins, lipids, carbohydrates, and nucleic acids) from smaller components. These reactions are responsible for the release of energy from chemical bonds and the storage of this energy. Metabolic reactions are also responsible for the transduction and transmission of information (for example, via the generation of cGMP as a secondary messenger or mRNA as a substrate for protein translation).

19.1.2 Why Model Metabolism?

An important application of metabolic modeling is in the prediction of drug effects. An important subject of modeling is the organism Mycobacterium tuberculosis. The disruption of the mycolic acid synthesis pathways of this organism can help control TB infection. Computational modeling gives us a platform for identifying the best drug targets in this system. Gene knockout studies in Escherichia coli have allowed scientists to determine which genes and gene combinations affect the growth of this important model organism. Both agreements and disagreements between models and experimental data can help us assess our knowledge of biological systems and help us improve our predictions about metabolic capabilities. In the next lecture, we will learn the importance of incorporating expression data into metabolic models. In addition, a variety of infectious disease processes involve metabolic changes at the microbial level.

19.2 Model Building

An overarching goal of metabolic modeling is the ability to take a picture modeling a pathway and change that into a mathematical formula modeling the pathway. For example, converting the following pathway into a mathematical model would be incredible useful.

19.2.1 Chemical Reactions

In metabolic models, we are concerned with modeling chemical reactions that are catalyzed by enzymes. Enzymes work by acting on a transition state of the enzyme-substrate complex that lowers the activation energy of a chemical reaction. The diagram on slide 5 of page 1 of the lecture slides demonstrates this phenomenon. A typical rate equation (which describes the conversion of the substrates S of the enzyme reaction into its products P) can be described by a Michaelis-Menten rate law:

\[
\frac{V}{V_{\text{max}}} = \frac{[S]}{K_m + [S]}
\]

In this equation, \( V \) is the rate of the equation as a function of substrate concentration \([S]\). It is clear that the parameters \( K_m \) and \( V_{\text{max}} \) are necessary to characterize the equation. The inclusion of multiple substrates, products, and regulatory relationships quickly increases the number of parameters necessary to characterize such an equation. The figures on slides 1, 2, and 3 of page 2 of the lecture notes demonstrate the complexity of biochemical pathways. Kinetic modeling quickly becomes infeasible because the necessary parameters are difficult to measure and also vary across organisms. Thus, we are interested in a modeling method that would allow us to avoid the use of large numbers of poorly-determined parameters. To this end, we recall the basic machinery of stoichiometry from general chemistry. Consider the chemical equation \( A + 2B \rightarrow 3C \), which says that one unit of reactant A combines with 2 units of reactant B to form 3 units.
of reactant C. The rate of formation of the compound X is given by the time derivative of [X]. Note that C forms three times as fast as A. Therefore, due to the stoichiometry of the reaction, we see that the reaction rate (or reaction flux) is given by

$$\text{flux} = \frac{d[A]}{dt} = \frac{1}{2} \frac{d[B]}{dt} = \frac{1}{3} \frac{d[C]}{dt}$$

### 19.2.2 Steady-State Assumption

The steady state assumption allows us to represent reactions entirely in terms of their chemistry (i.e. the stoichiometric relationships between the components of the enzymatic reaction) by assuming that there is no accumulation of any metabolite in the system. Note that this does not imply the absence of flux through any given reaction; rather, this assumption says that a single metabolite will not accumulate over a long period of time. An analogy is a series of waterfalls that contribute water to pools. As the water falls from one pool to another, the water levels do not change even though water continues to flow (see page 2 slide 5). This framework prevents us from being hindered by the overly complicated transient kinetics that can result from perturbations of the system. Since we are usually interested in long-term metabolic capabilities (functions on a scale longer than milliseconds or seconds), the steady state dynamics may give us all the information that we need.

An important consequence of the steady-state assumption is that reaction stoichiometries are conserved across species, whereas the biology of enzyme catalysis (and the parameters that characterize it) are not similarly conserved. Indeed, species-dependent parameters such as the activation energy of a reaction, substrate affinity of an enzyme, and the rate constants for various reactions are not required for steady-state
modeling. This makes the ability to generalize across species and reuse conserved pathways in models much more feasible.

19.2.3 Reconstructing Metabolic Pathways

There are several databases that can provide the information necessary to reconstruct metabolic pathways in silico. These databases allow reaction stoichiometry to be accessed using Enzyme Commission numbers, which are the same in each organism that utilizes the particular enzyme. Among the databases of interest are ExPASy, MetaCyc, and KEGG. These databases often contain pathways organized by function that can be downloaded in SBML format, making pathway reconstruction very easy for well-characterized pathways.

19.3 Metabolic Flux Analysis

Metabolic flux analysis (MFA) is a way of computing the distribution of reaction fluxes that is possible in a given metabolic network at steady state. We can place constraints on certain fluxes in order to limit the space described by the distribution of possible fluxes. In this section, we will develop a mathematical formulation for MFA. Once again, this analysis is independent of the particular biology of the system; rather, it will only depend on the (universal) stoichiometries of the reactions in question.

19.3.1 Mathematical Representation

Consider a system with \( m \) metabolites and \( n \) reactions. Let \( x_i \) be the concentration of substrate \( i \), so that the rate of change of the substrate concentration is given by the time derivative of \( x_i \). Let \( x \) be the column vector (with \( m \) components) with elements \( x_i \). For simplicity, we consider a system with \( m = 4 \) metabolites A, B, C, and D. This system will consist of many reactions between these metabolites, resulting in a complicated balance between these compounds.

Once again, consider the simple reaction \( A + 2B \rightarrow 3C \). We can represent this reaction in vector form as \((-1 -2 3 0)\). Note that the first two metabolites (A and B) have negative signs, since they are consumed in the reaction. Moreover, the elements of the vector are determined by the stoichiometry of the reaction, as in Section 2.1. We repeat this procedure for each reaction in the system. These vectors become the columns of the stoichiometric matrix \( S \). If the system has \( m \) metabolites and \( n \) reactions, \( S \) will be a \( m \times n \) matrix. Therefore, if we define \( v \) to be the \( n \)-component column vector of fluxes in each reaction, the vector \( Sv \) describes the rate of change of the concentration of each metabolite. Mathematically, this can be represented as the fundamental equation of metabolic flux analysis:

\[
\frac{dx}{dt} = Sv
\]

The matrix \( S \) is an extraordinarily powerful data structure that can represent a variety of possible scenarios in biological systems. For example, if two columns \( c \) and \( d \) of \( S \) have the property that \( c = d \), the columns represent a reversible reaction. Moreover, if a column has the property that only one component is nonzero, it represents in exchange reaction, in which there is a flux into (or from) a supposedly infinite sink (or source), depending on the sign of the nonzero component.

We now impose the steady state assumption, which says that the left size of the above equation is identically zero. Therefore, we need to find vectors \( v \) that satisfy the criterion \( Sv = 0 \). Solutions to this equation will determine feasible fluxes for this system.

19.3.2 Null Space of \( S \)

The feasible flux space of the reactions in the model system is defined by the null space of \( S \), as seen above. Recall from elementary linear algebra that the null space of a matrix is a vector space; that is, given two vectors \( y \) and \( z \) in the nullspace, the vector \( ay + bz \) (for real numbers \( a \), \( b \)) is also in the null space. Since the null space is a vector space, there exists a basis \( b_i \), a set of vectors that is linearly independent and spans the null space. The basis has the property that for any flux \( v \) in the null space of \( S \), there exist real numbers \( \alpha_i \) such that
How do we find a basis for the null space of a matrix? A useful tool is the singular value decomposition (SVD) \[ \text{SVD} \]. The singular value decomposition of a matrix S is defined as a representation \( S = U E V^* \), where \( U \) is a unitary matrix of size \( m \), \( V \) is a unitary matrix of size \( n \), and \( E \) is an \( m \times n \) diagonal matrix, with the (necessarily positive) singular values of \( S \) in descending order. (Recall that a unitary matrix is a matrix with orthonormal columns and rows, i.e. \( U^* U = U U^* = I \) the identity matrix). It can be shown that any matrix has an SVD. Note that the SVD can be rearranged into the equation \( S v = \sigma u \), where \( u \) and \( v \) are columns of the matrices \( U \) and \( V \) and \( \sigma \) is a singular value. Therefore, if \( \sigma = 0 \), \( v \) belongs to the null space of \( S \). Indeed, the columns of \( V \) that correspond to the zero singular values form an orthonormal basis for the null space of \( S \). In this manner, the SVD allows us to completely characterize the possible fluxes for the system.

19.3.3 Constraining the Flux Space

The first constraint mentioned above is that all steady-state flux vectors must be in the null space. Also negative fluxes are not thermodynamically possible. Therefore a fundamental constraint is that all fluxes must be positive. (Within this framework we represent reversible reactions as separate reactions in the stoichiometric matrix \( S \) having two unidirectional fluxes.)

These two key constraints form a system that can be solved by convex analysis. The solution region can be described by a unique set of Extreme Pathways, i.e. the steady state flux vectors \( v \) can be described as a positive linear combination of these extreme pathways. The Extreme Pathways, represented in slide 25 as vectors \( b_i \), circumscribe a convex flux cone. Each dimension is a rate for some reaction. In slide 25, the \( z \)-dimension represents the rate of reaction for \( v_3 \). We can recognize that at any point in time, the organism is living at a point in the flux cone, i.e. is demonstrating one particular flux distribution. Every point in the flux cone can be described by a possible steady state flux vector, while points outside the cone cannot.

One problem is that the flux cone goes out to infinity. Therefore an additional constraint is capping the flux cone by determining the maximum fluxes of any of our reactions (these values correspond to our \( V_{\text{max}} \) parameters). You do not need to determine all of the maximum flux vectors, only enough to cap the cone.

We can also add input and output fluxes that represent transport into and out of our cells (\( V_{\text{in}} \) and \( V_{\text{out}} \)), which are often much easier to measure than internal fluxes and can thus serve to help us to generate a more biologically relevant flux space. An example of an algorithm for solving this problem is the simplex algorithm \[ \text{Simplex} \]. Slides 24-27 demonstrate how constraints on the fluxes change the geometry of the flux cone. In reality, we are dealing with problems in higher dimensional spaces.

19.3.4 Linear Programming

Linear programming is a generic solution that is capable of solving optimization problems given linear constraints. These can be represented in a few different forms.
Canonical Form:

- Maximize: $c^T x$
- Subject to: $Ax \leq b$

Standard Form:

- Maximize $\sum c_i x_i$
- Subject to $a_{ij} X_i \leq b_i$ for all $i, j$
- Non-negativity constraint: $X_i \geq 0$

A concise and clear introduction to Linear Programming is available here: [http://www.purplemath.com/modules/linprog.htm](http://www.purplemath.com/modules/linprog.htm) The constraints described throughout section 3 give us the linear programming problem described in lecture. Linear programming can be considered a first approximation and is a classic problem in optimization. In order to try and narrow down our feasible flux, we assume that there exists a fitness function which is a linear combination of any number of the fluxes in the system. Linear programming (or linear optimization) involves maximizing or minimizing a linear function over a convex polyhedron specified by linear and non-negativity constraints.

![Figure 19.3: Maximizing two functions with linear programming.](figure)

We solve this problem by identifying the flux distribution that maximizes an objective function: The key point in linear programming is that our solutions lie at the boundaries of the permissible flux space and can be on points, edges, or both. By definition however, an optimal solution (if one exists) will lie at a point of the permissible flux space. This concept is demonstrated on slide 30. In that slide, $A$ is the stoichiometric matrix, $x$ is the vector of fluxes, and $b$ is a vector of maximal permissible fluxes.

Linear programs, when solved by hand, are generally done by the Simplex method. The simplex method sets up the problem in a matrix and performs a series of pivots, based on the basic variables of the problem statement. In worst case, however, this can run in exponential time. Luckily, if a computer is available, two other algorithms are available. The ellipsoid algorithm and Interior Point methods are both capable of solving any linear program in polynomial time. It is interesting to note, that many seemingly difficult problems can be modeled as linear programs and solved efficiently (or as efficiently as a generic solution can solve a specific problem).

In microbes such as E. coli, this objective function is often a combination of fluxes that contributes to biomass, as seen in slide 31. However, this function need not be completely biologically meaningful. For example, we might simulate the maximization of mycolates in *M. tuberculosis*, even though this isn't happening biologically. It would give us meaningful predictions about what perturbations could be performed in vitro that would perturb mycolate synthesis even in the absence of the maximization of the production of those metabolites. Flux balance analysis (FBA) was pioneered by Palsson's group at UCSD and has since been applied to E. coli, M. tuberculosis, and the human red blood cell [? ].
19.4 Applications

19.4.1 In Silico Detection Analysis

With the availability of such a powerful tool like FBA, more questions naturally arise. For example, are we able to predict gene knockout phenotype based on their simulated effects on metabolism? Also, why would we try to do this, even though other methods, like protein interaction map connective, exist? Such analysis is actually necessary, since other methods do not take into direct consideration the metabolic flux or other specific metabolic conditions.

Knocking out a gene in an experiment is simply modeled as removing one of the columns (reactions) from the stoichiometric matrix. (A question during class clarified that a single gene can knock out multiple columns/reactions.) Thereby, these knockout mutations will further constrain the feasible solution space by removing fluxes and their related extreme pathways. If the original optimal flux was outside is outside the new space, then new optimal flux is created. Thus the FBA analysis will produce different solutions. The solution is a max growth rate, which may be confirmed or disproven experimentally. The growth rate at the new solution provides a measure of the knockout phenotype. If these gene knockouts are in fact lethal, then the optimal solution will be a growth rate of zero.

Figure 19.4: Removing a reaction is the same as removing a gene from the stoichiometric matrix.

Figure 19.5: Constraining the feasible solution space may create a new optimal flux.
Studies by Edwards, Palsson (1900) explore knockout phenotype prediction use to predict metabolic changes in response to knocking out enzymes in E. coli, a prokaryote [? ]. In other words, an in silico metabolic model of E. coli was constructed to simulate mutations affecting the glycolysis, pentose phosphate, TCA, and electron transport pathways (436 metabolites and 719 reactions included). For each specific condition, the optimal growth of mutants was compared to non-mutants. The in vivo and in silico results were then compared, with 86% agreement. The errors in the model indicate an underdeveloped model (lack of knowledge). The authors discuss 7 errors not modeled by FBA, including mutants inhibiting stable RNA synthesis and producing toxic intermediates.

19.4.2 Quantitative Flux In Silico Model Predictions

Can models quantitatively predict fluxes, growth rate? We demonstrate the ability of FBA to give quantitative predictions about growth rate and reaction fluxes given different environmental conditions. More specifically, prediction refers to externally measurable fluxes as a function of controlled uptake rates and environmental conditions. Since FBA maximizes an objective function, resulting in a specific value for this function, we should in theory be able to extract quantitative information from the model.

An early example by Edwards, Ibarra, and Palsson (1901), predicted the growth rate of E. coli in culture given a range of fixed uptake rates of oxygen and two carbon sources (acetate and succinate), which they could control in a batch reactor [? ]. They assumed that E. coli cells adjust their metabolism to maximize growth (using a growth objective function) under given environmental conditions and used FBA to model the metabolic pathways in the bacterium. The input to this particular model is acetate and oxygen, which is labeled as $V_{IN}$.

The controlled uptake rates fixed the values of the oxygen and acetate/succinate input fluxes into the network, but the other fluxes were calculated to maximize the value of the growth objective.

The growth rate is still treated as the solution to the FBA analysis. In sum, optimal growth rate is predicted as a function of uptake constraints on oxygen versus acetate and oxygen versus succinate. The basic model is a predictive line and may be confirmed in a bioreactor experimentally by measuring the uptake and growth from batch reactors (note: experimental uptake was not constrained, only measured).

This model by Palsson was the first good proof of principle in silico model. The authors quantitative growth rate predictions under the different conditions matched very closely to the experimentally observed growth rates, implying that E. coli do have a metabolic network that is designed to maximize growth. It had good true positive and true negative rates. The agreement between the predictions and experimental results is very impressive for a model that does not include any kinetic information, only stoichiometry. Prof. Galagan cautioned, however, that it is often difficult to know what good agreement is, because we don’t know the significance of the size of the residuals. The organism was grown on a number of different nutrients. Therefore, the investigators were able to predict condition specific growth. Keep in mind this worked, since only certain genes are necessary for some nutrients, like fbp for gluconeogenesis. Therefore, knocking out fbp will only be lethal when there is no glucose in the environment, a specific condition that resulted in a growth solution when analyzed by FBA.

19.4.3 Quasi Steady State Modeling (QSSM)

We’re now able describe how to use FBA to predict time-dependent changes in growth rates and metabolite concentrations using quasi steady state modeling. The previous example used FBA to make quantitative growth predictions under specific environmental conditions (point predictions). Now, after growth and uptake fluxes, we move on to another assumption and type of model.

Can we use a steady state model of metabolism to predict the time-dependent changes in the cell or environments? We do have to make a number of quasi steady state assumptions (QSSA):

1. The metabolism adjusts to the environmental/cellular changes more rapidly than the changes themselves
2. The cellular and environmental concentrations are dynamic, but metabolism operates on the condition that the concentration is static at each time point (steady state model).
Is it possible to use QSSM to predict metabolic dynamics over time? For example, if there is less acetate being taken in on a per cell basis as the culture grows, then the growth rate must slow. But now, QSSA assumptions are applied. That is, in effect, at any given point in time, the organism is in steady state.

What values does one get as a solution to the FBA problem? There are fluxes the growth rate. We are predicting rate and fluxes (solution) where VIN/OUT included. Up to now we assumed that the input and output are infinite sinks and sources. To model substrate/growth dynamics, the analysis is performed a bit differently from prior quantitative flux analysis. We first divide time into slices $\delta t$. At each time point $t$, we use FBA to predict cellular substrate uptake ($S_u$) and growth ($g$) during interval $\delta t$. The QSSA means these predictions are constant over $\delta t$. Then we integrate to get the biomass ($B$) and substrate concentration ($S_c$) at the next time point $t + \delta t$. Therefore, the new VIN is calculated each time based on points $\delta t$ in-between time. Thus we can predict the growth rate and glucose and acetate uptake (nutrients available in the environment). The four step analysis is:

1. The concentration at time $t$ is given by the substrate concentration from the last step plus any additional substrate provided to the cell culture by an inflow, such as in a fed batch.
2. The substrate concentration is scaled for time and biomass ($X$) to determine the substrate availability to the cells. This can exceed the maximum uptake rate of the cells or be less than that number.
3. Use the flux balance model to evaluate the actual substrate uptake rate, which may be more or less than the substrate available as determined by step 2.
4. The concentration for the next time step is then calculated by integrating the standard differential equations:

$$\frac{dB}{dt} = gB \rightarrow B = B_0 e^{gt}$$

$$\frac{dS_c}{dt} = -S_u B \rightarrow S_c = S_c^0 X (e^{gt} - 1)$$

The additional work by Varma et al. (1994) specifies the glucose uptake rate a priori. The model simulations work to predict time-dependent changes in growth, oxygen uptake, and acetate secretion. This converse model plots uptake rates versus growth, while still achieving comparable results in vivo and in silico. The researchers used quasi steady state modeling to predict the time-dependent profiles of cell growth and metabolite concentrations in batch cultures of E. coli that had either a limited initial supply of glucose (left) or a slow continuous glucose supply (right diagram). A great fit is evident.

The diagrams above show the results of the model predictions (solid lines) and compare it to the experimental results (individual points). Thus, in E. coli, quasi steady state predictions are impressively accurate even with a model that does not account for any changes in enzyme expression levels over time. However, this model would not be adequate to describe behavior that is known to involve gene regulation. For example, if the cells had been grown on half-glucose/half-lactose medium, the model would not have been able to predict the switch in consumption from one carbon source to another. (This does occur experimentally when E. coli activates alternate carbon utilization pathways only in the absence of glucose.)

### 19.4.4 Regulation via Boolean Logic

There is a number of levels of regulation through which metabolic flux is controlled at the metabolite, transcriptional, translational, post-translational levels. FBA associated errors may be explained by incorporation of gene regulatory information into the models. One way to do this is Boolean logic. The following table describes if genes for associated enzymes are on or off in presence of certain nutrients (an example of incorporating E. coli preferences mentioned above):

<table>
<thead>
<tr>
<th>Nutrient Status</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>no glucose(0)</td>
<td>ON</td>
</tr>
<tr>
<td>glucose present(1)</td>
<td>OFF</td>
</tr>
<tr>
<td>acetate present(1)</td>
<td>ON</td>
</tr>
<tr>
<td>acetate present(1)</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Therefore, one may think that the next step to take is to incorporate this fact into the models. For example, if we have glucose in the environment, the acetate processing related genes are off and therefore absent from the S matrix which now becomes dynamic as a result of incorporation of regulation into our model. In the end, our model is not quantitative. The basic regulation then describes that if one nutrient-processing enzyme is on, the other is off. Basically it is a bunch of Boolean logic, based on presence of enzymes, metabolites, genes, etc. These Boolean style assumptions are then used at every small change in time $dt$ to evaluate the growth rate, the fluxes, and such variables. Then, given the predicted fluxes, the $V_{IN}$, the $V_{OUT}$, and the system states, one can use logic to turn genes off and on, effectively a $\delta S$ per $\delta t$. We can start putting together all of the above analyses and come up with a general approach in metabolic modeling. We can tell that if glycolysis is on, then gluconeogenesis must be off.

The first attempt to include regulation in an FBA model was published by Covert, Schilling, and Palsson in 1901 [?]. The researchers incorporated a set of known transcriptional regulatory events into their analysis of a metabolic regulatory network by approximating gene regulation as a Boolean process. A reaction was said to occur or not depending on the presence of both the enzyme and the substrate(s): if either the enzyme that catalyzes the reaction (E) is not expressed or a substrate (A) is not available, the reaction flux will be zero:

$$\text{rxn} = \text{IF} \ (A) \ \text{AND} \ (E)$$

Similar Boolean logic determined whether enzymes were expressed or not, depending on the currently expressed genes and the current environmental conditions. For example, transcription of the enzyme (E) occurs only if the appropriate gene (G) is available for transcription and no repressor (B) is present:

$$\text{trans} = \text{IF} \ (G) \ \text{AND NOT} \ (B)$$

The authors used these principles to design a Boolean network that inputs the current state of all relevant genes (on or off) and the current state of all metabolites (present or not present), and outputs a binary vector containing the new state of each of these genes and metabolites. The rules of the Boolean network were constructed based on experimentally determined cellular regulatory events. Treating reactions and enzyme/metabolite concentrations as binary variables does not allow for quantitative analysis, but this method can predict qualitative shifts in metabolic fluxes when merged with FBA. Whenever an enzyme is absent, the corresponding column is removed from the FBA reaction matrix, as was described above for knockout phenotype prediction. This leads to an iterative process:

1. Given the initial states of all genes and metabolites, calculate the new states using the Boolean network;
2. perform FBA with appropriate columns deleted from the matrix, based on the states of the enzymes, to determine the new metabolite concentrations;
3. repeat the Boolean network calculation with the new metabolite concentrations; etc. The above model is not quantitative, but rather a pure simulation of turning genes on and off at any particular time instant.

On a few metabolic reactions, there are rules about allowing organism to shift carbon sources (C1, C2).

An application of this method from the study by Covert et al.[?] was to simulate diauxic shift, a shift from metabolizing a preferred carbon source to another carbon source when the preferred source is not available. The modeled process includes two gene products, a regulatory protein RpC1, which senses (is activated by) Carbon 1, and a transport protein Tc2, which transports Carbon 2. If RpC1 is activated by Carbon 1, Tc2 will not be transcribed, since the cell preferentially uses Carbon 1 as a carbon source. If Carbon 1 is not available, the cell will switch to metabolic pathways based on Carbon 2 and will turn on expression of Tc2.

The Booleans can represent this information:

$$\text{RpC1} = \text{IF(Carbon1)} \ \text{Tc2} = \text{IF NOT(RpC1)}$$

Covert et al. found that this approach gave predictions about metabolism that matched results from experimentally induced diauxic shift. This diauxic shift is well modeled by the in silico analysis see above
figure. In segment A, C1 is used up as a nutrient and there is growth. In segment B, there is no growth as C1 has run out and C2 processing enzymes are not yet made, since genes have not been turned on (or are in the process), thus the delay of constant amount of biomass. In segment C, enzymes for C2 turned on and the biomass increases as growth continues with a new nutrient source. Therefore, if there is no C1, C2 is used up. As C1 runs out, the organism shifts metabolic activity via genetic regulation and begins to take up C2. Regulation predicts diauxie, the use of C1 before C2. Without regulation, the system would grow on both C1 and C2 together to max biomass.

So far we have discussed using this combined FBA-Boolean network approach to model regulation at the transcriptional/translational level, and it will also work for other types of regulation. The main limitation is for slow forms of regulation, since this method assumes that regulatory steps are completed within a single time interval (because the Boolean calculation is done at each FBA time step and does not take into account previous states of the system). This is fine for any forms of regulation that act at least as fast as transcription/translation. For example, phosphorylation of enzymes (an enzyme activation process) is very fast and can be modeled by including the presence of a phosphorylase enzyme in the Boolean network. However, regulation that occurs over longer time scales, such as sequestration of mRNA, is not taken into account by this model. This approach also has a fundamental problem in that it does not allow actual experimental measurements of gene expression levels to be inputted at relevant time points.

But we don't need our simulations to artificially predict whether certain genes are on or off, but rather measure them. With microarray expression data, we can figure out what genes are on and off and therefore use this information in our models.

19.4.5 Coupling Gene Expression with Metabolism

But actually, we don't need to artificially model gene levels, we can measure them. As discussed previously in lecture, we can measure mRNA expression throughout an organism by microarray experiments. As this information provides a set of expression levels of different genes under a certain condition, it would be extremely useful to incorporate it into the FBA. Usually, data from microarray experiments is clustered, and unknown genes are hypothesized to have function similar to the function of those known genes with which they cluster. This analysis can be faulty, however, as genes with similar actions may not always cluster together. Incorporating microarray expression data into FBA could allow an alternate method of interpretation of the data. Here arises a question, what is the relationship between gene level and flux through a reaction?

Say the reaction $A \rightarrow B$ is catalyzed by an enzyme. If a lot of A present, increased expression of the gene for the enzyme causes increased reaction rate. Otherwise, increasing gene expression level will not increase reaction rate. However, the enzyme concentration can be treated as a constraint on the maximum possible flux, given that the substrate also has a reasonable physiological limit.

The next step, then, is to relate the mRNA expression level to the enzyme concentration. This is more difficult, since cells have a number of regulatory mechanisms to control protein concentrations independently of mRNA concentrations. For example, translated proteins may require an additional activation step (e.g. phosphorylation), each mRNA molecule may be translated into a variable number of proteins before it is degraded (e.g. by antisense RNAs), the rate of translation from mRNA into protein may be slower than the time intervals considered in each step of FBA, and the protein degradation rate may also be slow. Despite these complications, the mRNA expression levels from microarray experiments are usually taken as upper bounds on the possible enzyme concentrations at each measured time point. Given the above relationship between enzyme concentration and flux, this means that the mRNA expression levels are also upper bounds on the maximum possible fluxes through the reactions catalyzed by their encoded proteins. The validity of this assumption is still being debated, but it has already performed well in FBA analyses and is consistent with recent evidence that cells do control metabolic enzyme levels primarily by adjusting mRNA levels. (In 1907, Professor Galagan discussed a study by Zaslaver et al. (1904) that found that genes required in an amino acid biosynthesis pathway are transcribed sequentially as needed [? ]). This is a particularly useful assumption for including microarray expression data in FBA, since FBA makes use of maximum flux values to constrain the flux balance cone.

Colijn et al. address the question of algorithmic integration of expression data and metabolic networks [? ]. They apply FBA to model the maximum flux through each reaction in a metabolic network. For
example, if microarray data is available from an organism growing on glucose and from an organism growing on acetate, significant regulatory differences will likely be observed between the two datasets. $V_{\text{max}}$ tells us what the maximum we can reach. Microarray detects the level of transcripts, and it gives an upper boundary of $V_{\text{max}}$.

In addition to predicting metabolic pathways under different environmental conditions, FBA and microarray experiments can be combined to predict the state of a metabolic system under varying drug treatments. For example, several TB drugs target mycolic acid biosynthesis. Mycolic acid is a major cell wall constituent. In a 1904 paper by Boshoff et al., researchers tested 75 drugs, drug combinations, and growth conditions to see what effect different treatments had on mycolic acid synthesis. In 1905, Raman et al. published an FBA model of mycolic acid biosynthesis, consisting of 197 metabolites and 219 reactions.

The basic flow of the prediction was to take a control expression value and a treatment expression value for a particular set of genes, then feed this information into the FBA and measure the final effect on the treatment on the production of mycolic acid. To examine predicted inhibitors and enhancers, they examined significance, which examines whether the effect is due to noise, and specificity, which examines whether the effect is due to mycolic acid or overall suppression/enhancement of metabolism. The results were fairly encouraging. Several known mycolic acid inhibitors were identified by the FBA. Interesting results were also found among drugs not specifically known to inhibit mycolic acid synthesis. 4 novel inhibitors and 2 novel enhancers of mycolic acid synthesis were predicted. One particular drug, Triclosan, appears to be an enhancer according to the FBA model, whereas it is currently known as an inhibitor. Further study of this particular drug would be interesting. Experimental testing and validation are currently in progress.

Clustering may also be ineffective in identifying function of various treatments. Predicted inhibitors, and predicted enhancers of mycolic acid synthesis are not clustered together. In addition, no labeled training set is required for FBA-based algorithmic classification, whereas it is necessary for supervised clustering algorithms.

19.4.6 Predicting Nutrient Source

Now, we get the idea of predicting the nutrient source that an organism may be using in an environment, by looking at expression data and looking for associated nutrient processing gene expression. This is easier, since we can go into the environment and measure all chemical levels, but we can get expression data rather easily. That is, we try to predict a nutrient source through predictions of metabolic state from expression data, based on the assumption that organisms are likely to adjust metabolic state to available nutrients. The nutrients may then be ranked by how well they match the metabolic states.

The other way around could work too. Can I predict a nutrient given a state? Such predictions could
be useful for determining the nutrient requirements of an organism with an unknown natural environment, or for determining how an organism changes its environment. (TB, for example, is able to live within the environment of a macrophage phagolysosome, presumably by altering the environmental conditions in the phagolysosome and preventing its maturation.)

We can use FBA to define a space of possible metabolic states and choose one. The basic steps are to:

- Start with max flux cone (representing best growth with all nutrients available in environment). Find optimal flux for each nutrient.
- Apply expression data set (still not knowing nutrient). This will allow you to constrain the cone shape and figure out the nutrient, which is represented as one with the closest distance to optimal solution.

In Figure 8, you may see that the first cone has a number of optimals, so the real nutrient is unknown. However, after expression data is applied, the cone is reshaped. It has only one optimal, which is still in feasible space and thereby must be that nutrient you are looking for.

As before, the measured expression levels provide constraints on the reaction fluxes, altering the shape of the flux-balance cone (now the expression-constrained flux balance cone). FBA can be used to determine
the optimal set of fluxes that maximize growth within these expression constraints, and this set of fluxes can be compared to experimentally-determined optimal growth patterns under each environmental condition of interest. The difference between the calculated state of the organism and the optimal state under each condition is a measure of how sub-optimal the current metabolic state of the organism would be if it were in fact growing under that condition.

Expression data from growth and metabolism may then be applied to predict the carbon source being used. For example, consider E. coli nutrient product. We can simulate this system for glucose versus acetate. The color indicates the distance from the constrained flux cone to the optimal flux solution for that nutrient combo (same procedure described above). Then, multiple nutrients may be ranked, prioritized according to expression data. Unpublished data from Desmond Lun and Aaron Brandes provide an example of this approach.

They used FBA to predict which nutrient source E. coli cultures were growing on, based on gene expression data. They compared the known optimal fluxes (the optimal point in flux space) for each nutrient condition to the allowed optimal flux values within the expression-constrained flux-balance cone. Those nutrient conditions with optimal fluxes that remained within (or closest to) the expression-constrained cone were the most likely possibilities for the actual environment of the culture.

Results of the experiment are shown in Figure 9, where each square in the results matrices is colored based on the distance between the optimal fluxes for that nutrient condition and the calculated optimal fluxes based on the expression data. Red values indicate large distances from the expression-constrained flux cone and blue values indicate short distances from the cone. In the glucose-acetate experiments, for example, the results of the experiment on the left indicate that low acetate conditions are the most likely (and glucose was the nutrient in the culture) and the results of the experiment on the right indicate that low glucose/medium acetate conditions are the most likely (and acetate was the nutrient in the culture). When 6 possible nutrients were considered, the model always predicted the correct one, and when 18 possible nutrients were considered, the correct one was always one of the top 4 ranking predictions. These results suggest that it is possible to use expression data and FBA modeling to predict environmental conditions from information about the metabolic state of an organism.

This is important because TB uses fatty acids in macrophages in immune systems. We do not know which ones exactly are utilized. We can figure out what the TB sees in its environment as a food source and proliferation factor by analyzing what related nutrient processing genes are turned on at growth phases and such. Thereby we can figure out the nutrients it needs to grow, allowing for a potential way to kill it off by not supplying such nutrients or knocking out those particular genes.

It is easier to get expression data to see flux activity than see what is being used up in the environment by analyzing the chemistry on such a tiny level. Also, we might not be able to grow some bacteria in lab, but we can solve the problem by getting the expression data from the bacteria growing in a natural environment and then seeing what it is using to grow. Then, we can add it to the laboratory medium to grow the bacteria successfully.
19.5 Current Research Directions

19.6 Further Reading

  - If gene expression lower than some threshold, turn the gene off in the model.
  - Nested optimization problem.
  - First, standard FBA
  - Second, maximize the number of enzymes whose predicted flux activity is consistent with their measured expression level

19.7 Tools and Techniques

- Kegg
- BioCyc
- Pathway Explorer (pathwayexplorer.genome.tugraz.at)
- Palssons group at UCSD (http://gcrg.ucsd.edu/)
- www.systems-biology.org
- Biomodels database (www.ebi.ac.uk/biomodels/)
- JWS Model Database (jjj.biochem.sun.ac.za/database/index.html)

19.8 What Have We Learned?

Bibliography


Part IV

Phylogenomics and Population Genomics
List of figures

Common Phylogenetic Tree Terminology ??
Three Types of Trees: Cladogram, Phylogram, Ultrametric ??
Markov Model to Correct for Back Mutations ??
Jukes-Cantor Model ??
Kimura Model ??
Ultrametric Distances ??
Additive Distances ??
20.1 Introduction

Continuing on the theme of evolution from the previous lecture, we look at how to build evolutionary trees from data. The history of the evolution of life can be depicted as a tree, with the extant species being the leaves of the tree. All extant species, even those from different kingdoms, can be traced back to a common ancestor, implying the tree structure in the evolutionary history of life. We can place more than just organisms into trees: individual genes from the genomes of various organisms can also be placed into trees, and from these gene phylogenies, we can often infer the species phylogenies. Often, this is done with the geological record in mind: not all branches of the tree of life terminate in the present there are many that end in the various mass extinctions that have occurred in the past (e.g. K-T extinction of dinosaurs).

The general problem of phylogenetics would then be to infer the complete ancestry (i.e. the tree, or phylogeny) of a set of objects, given a set of traits for each object. The objects themselves can be biological (species, genes, cell types, diseases, cancers), or non-biological (languages, religious faiths, cars, architectural styles). Individual objects have a set of traits that can be defined by morphology, gene expression, sequence, motifs, biochemical and molecular characteristics words etc. We can use the historical record to infer some of the branch points (i.e. nodes) of the trees, such as fossils, imprints, geological events, living fossils (species that have evolved little), sequencing extinct species, historical records (oral or written). We will focus on genes for the purposes of the lecture / class.

It is useful to keep the concepts of homology in mind: homology being genes related by common descent: they are subsequently paralogous if they arose from a gene duplication event, or orthologs if they arose from a speciation event.

**FAQ**

**Q:** Would it be possible to use extinct species’ DNA sequences?

**A:** Current technologies only allow for usage of extant sequences. However, there have been a few successes in using extinct species’ DNA. DNA from frozen mammoths have been collected and are being sequenced but due to DNA breaking down over time and contamination from the environment, it is very hard to extract correct sequences.

20.2 Basics of Phylogeny

Traditionally, phylogeny of species was done using differences in morphology of the species, using different arbitrary measurements of various body parts. This resulted in a relatively small number of traits, but the traits are (chosen to be) well-behaved (i.e. they arose only once). One would then use these sets of traits to derive a tree by parsimony, using Occam’s razor, where the least complex explanation is most likely to be correct. This, however, is prone to errors as traits are not necessarily well-behaved (e.g. they may have arisen more than once such as through convergent evolution).

Modern methods use the genomes / individual gene sequences of various species. Now, instead of a few well-behaved sets of traits, we have many traits (as many as the length of a gene / genome). Furthermore, the traits are no longer well-behaved we have both back mutations (reverting back to the previous form), as well as convergent evolution (two different species having the same trait with no common history of that trait).

Depicting a tree of branching nodes with leaves on one side and the root on the other (like above), the branch lengths in a tree can mean different things. There are three general ways of defining branch lengths in phylogenetic trees:

**Cladogram:** gives no meaning to branch lengths; only the sequence and topology of the branching matters.

**Phylogram:** Branch lengths are directly related to the amount of genetic change. The longer the branch of a tree, the greater the amount of phylogenetic change that has taken place. The leaves in this tree may not necessarily end on the same vertical line, due to different rates of mutation.
Figure 20.1: Defining tree terminology. A tree of branching nodes is depicted with leaves on one side and the root on the other.

**Chronogram (ultrametric tree):** Branch lengths are directly related to time. The longer the branches of a tree, the greater the amount of time that has passed. The leaves in this tree necessarily end on the same vertical line (i.e. they are the same distance from the root), since they are all in the present unless extinct species were included in the tree. Although there is a correlation between branch lengths and genetic distance on a chronogram, they are not necessarily exactly proportional because evolution rates / mutation rates were not necessarily the same for different species at different time periods.

![Common Phylogenetic Tree Terminology](image)

Figure 20.2: Three types of trees.

As computational biologists, our goal would be to use modern phylogenetic methods to go from the various nucleotide and/or peptide alignment data that we learn from sequencing to recreate the evolutionary history of those sequences into a tree. Typically, these trees are binary trees (each non-leaf node splits into exactly two branches), however, there is evidence for trinary and higher trees. Algorithms to create those trees are still being actively researched, and the most modern methods can take in metadata such as constraints from known phylogenies as additional inputs for building trees.

There are two main approaches to building trees from the sequence alignment data: distance-based methods, and character-based methods.
Distance based methods convert the individual pairwise alignments into distances (reduces the dimension of the problem), and then builds a tree based on those distances. The algorithms used to build the trees from the distances can be simple (e.g. hierarchical clustering, which we have gone through previously TODO: missing cite the previous lecture where hierarchal clustering was discussed, or more complex, as we shall see later.

Character-based methods use all the available information at every step of tree-building. This is usually done as iterations between tree-proposal and tree scoring, finding the best trees that either are the most likely (maximum likelihood), or require the least number of events (parsimony).

FAQ

Q: Is there more information than distance between the leaves on a tree? (clarification on what happens when one converts individual pairwise alignments to distances)

A: Trees represent series of events distance is not enough to account for different rates of evolution. Essentially a tree is a lower dimensional projection of high dimensional space

FAQ

Q: Picture interpretation on slide (the slide titled “Two kinds of phylogenetic methods”)

A: Rows are genes, alignment on columns, as is the typical view.

FAQ

Q: During tree comparison how do we score trees if we dont have the correct answer?

A: The correct trees can only come from external sources (based on traditional methods). Can verify that gene tree matches known species trees, can use training sets vs testing sets etc.

It is possible that different gene trees will yield different species trees, even if the genes are from the same set of species, implying that there might be different evolution rates for genes within a species.

20.3 Alignment to Distances

How do we convert pairwise sequence alignment to pairwise distances? This has to be based on an evolutionary model, and we derive the distances consistently based on the model. We have various models for this, with varying complexity (increasing in the order listed).
1. We can use absolute differences between sequences to derive distance. This assumes a uniform rate of divergence (1 parameter).

2. We can model transitions and transversions with different relative contribution to distances (2 parameters).

3. We can model mutations that are synonymous and non-synonymous, i.e. whether the substitutions changed the resulting amino acid in the protein (i.e. Ka and Ks rates) (multiple parameters).

In all cases, we need to also account for back mutations, as it is possible that some regions have had multiple mutations (which we would only have seen as a single mutation, or even no mutation). i.e. the number of observed substitutions ≠ number of actual mutation events. We can correct for this using a Markov model:

![Markov model](image)

Which simplifies to:

![Simplified Markov model](image)

The subsequent derivations for the various models are best viewed on the lecture slides TODO: missing @scribe: Insert images from the slides that the subsequent descriptions refer to. The following are notes on the models:

**Jukes-Cantor model** is a 1-parameter model, with a fixed probability of mutating (regardless of destination). An example of a Jukes-Cantor model is

![Jukes-Cantor model](image)

We might also like to consider (for statistical purposes) for a given observed distance, how confident am I of actual distance? The Jukes-Cantor model, however, is a rough estimate because it does not account for transitions/transversions. The **Kimura model** builds on the Jukes-Cantor model, accounting for transitions/transversions.

We can also account for unequal base frequencies; various models have been developed for that. This is a better model than basic Jukes-Cantor.
Figure 20.5: The Kimura model builds on the Jukes-Cantor model, accounting for transitions, transversions, and unequal base frequencies.

**FAQ**

Q: Can we use different parameters for different parts of the tree? To account for different mutation rates?

A: It's possible; it is a current area of research.

### 20.4 Distance to Trees

Creating trees from derived distances as calculated in the previous section is the second step in building a tree. We should be aware of the duality of tree-space vs distance space: a distance matrix can be mapped to a tree in an unconstrained manner, but a tree maps onto a distance matrix in a constrained manner. i.e. once we have built a tree the path between any two leaves of the tree is constrained to the branches of that tree. Hence, in tree building, we aim to minimize discrepancy between observed distances and tree-based distances, particularly in a noisy environment where distances implied by the tree do not necessarily match the derived distances.

The nature of our distance matrix can help us determine what tree-building algorithms to use.

**Ultrametric:** Ultrametric: This means that for all points \(i, j, k\), 2 distances are equal, third is smaller. This implies that all paths from leaves are equidistant to the root, and solving this tree is trivial. If your distance matrix was generated by an ultrametric tree, then hierarchical clustering (UPGMA) is guaranteed to find the right tree.

![Ultrametric tree](image)

**Figure 20.6: Ultrametric distances.**

Additive. This means that for all points \(i, j, k, l\), let

\[
\begin{align*}
d(i, j) &= a + b; \\
d(k, l) &= c + d; \\
m &\text{ exists such that:} \\
d(i, k) &= a + m + c; \\
d(j, l) &= b + m + d.
\end{align*}
\]
This implies that all pairwise distances are obtained from a tree. Everything else would be the general case, which is the most likely scenario. This can be due to noise (when our parameters for our evolutionary models are not precise), stochasticity and randomness (due to small samples), fluctuations, different rates of mutations, gene conversions and horizontal transfer. Because of this, we need tree-building algorithms that are able to handle noisy distance matrices.

**FAQ**

**Q:** In Figure ?? The m and r sequence divergence metrics can have some overlap so distance between mouse and rat is not simply m+r. Wouldnt that only be the case if there was no overlap?

**A:** If you model evolution correctly, then you would get evolutionary distance. Its an inequality rather than an equality and I agree that you cant exactly infer that the given distance is the precise distance. Therefore, the sequences distance between mouse and rat is probably less than m + r because of overlap, convergent evolution, and transversions.

### 20.4.1 Possible Theoretical and Practical Issues with Discussed Approach

A special point must be made about distances. Since distances are typically calculated between aligned gene sequences, most current tree reconstruction methods rely on heavily conserved genes, as non-conserved genes would not give information on species without those genes. This causes the ignoring of otherwise useful data. Therefore, there are some algorithms that try to take into account less conserved genes in reconstructing trees but these algorithms tend to take a long time due to the NP-Hard nature of reconstructing trees.

Additionally, aligned sequences are still not explicit in regards to the events that created them. That is, combinations of speciation, duplication, loss, and horizontal gene transfer (hgt) events are easy to mix up because only current DNA sequences are available. (see ?? for a commentary on such theoretical issues) A duplication followed by a loss would be very hard to detect. Additionally, a duplication followed
by a speciation could look like an HGT event. Even the probabilities of events happening is still contested, especially horizontal gene transfer events.

Another issue is that often multiple marker sequences are concatenated and the concatenated sequence is used to calculate distance and create trees. However, this approach assumes that all the concatenated genes had the same history and there is debate over if this is a valid approach given that events such as hgt and duplications as described above could have occurred differently for different genes. [? ] is an article showing how different phylogenetic relationships were found depending on if the tree was created using multiple genes concatenated together or if it was created using each of the individual genes. Conversely, additional [? ] claims that while hgt is prevalent, orthologs used for phylogenetic reconstruction are consistent with a single tree of life. These two issues indicate that there is clearly debate in the field on a non arbitrary way to define species and to infer phylogenetic relationships to recreate the tree of life.

20.5 Current Research Directions

20.6 Further Reading

20.7 Tools and Techniques

20.7.1 Project Ideas

1. Creating better distance models such as taking into account duplicate genes or loss of genes. It may also be possible to analyze sequences for peptide coding regions and calculate distances based on peptide chains too.

2. Creating a faster/more accurate search algorithm for turning distances into trees.

3. Analyze sequences to calculate probabilities of speciation, duplication, loss, and horizontal gene transfer events.

4. Extending an algorithm that looks for HGTs to look for extinct species. A possible use for HGTs is that if a program were to infer HGTs between different times, it could mean that there was a speciation where one branch is now extinct (or not yet discovered) and that branch had caused an HGT to the other extant branch.

20.7.2 Project Datasets

1. 1000 Genomes Project http://www.1000genomes.org/

2. Microbes Online http://microbesonline.org/

20.8 What Have We Learned?

Bibliography

[1] 1000 genomes project.


CHAPTER

TWENTYONE

PHYLOGENOMICS II

Guest Lecture by
Matt Rasmussen
Scribed by Jerry Wang and Dhruv Garg

List of figures

Heuristic Tree Search ??
Heuristic Tree Scoring Metric ??
Site Evolution ??
Sequence Evolution ??
Sequence Evolution Over an Entire Tree ??
Peeling Algorithm ??
Gene Family Evolution ??
Maximum Parsimony Reconciliation (MPR) ??
MPR Recursive Algorithm ??
Species Trees to Improve Gene Tree Reconstruction ??
Developing a Rate Model ??
Modeling Branch Length as Two Rate Components ??
The Wright-Fisher Model. ??
Wright-Fisher Over Many Generations/ Ignoring the Order of Chromosomes ??
Coalescent Model ??
Probability Distribution for Coalescent Model ??
Multispecies Coalescent Model ??
MPR Reconciliation of Genes and Species Tree ??
Inaccuracies in the Gene Tree ??
21.1 Introduction

Guest lecturer Matt Rasmussen, a former student of Manoliis presented our second phylogenomics lecture. The lecture finished explaining maximum likelihood methods for phylogenetics, and then progressed to more advanced uses of phylogenetics such as inferring orthologs, paralogs, gene duplication and gene loss. This led to learning across gene trees and modeling populations and allele frequencies.

In previous lectures, we studied various algorithms to obtain phylogenetic species trees. Similar studies can be performed to study phylogeny of gene families, or sets of orthologous and paralogous genes. Given multiply aligned sequences, several techniques discussed in previous lectures could be employed for constructing a gene tree, including nearest neighbor joining, and hierarchical clustering. If in addition to the aligned genes, we also have a species tree (which can often be taken as a given for sufficiently diverged species), then we should be able to formulate a consistent view of the evolutionary process; namely, we hope to map the gene tree onto the species tree. These mappings between the two trees are called reconciliations. The standard phylogenomic pipeline can be summarized as follows:

1. Blast protein sequences against each other to score similarities.
2. Use this metric to cluster genes into families of relatedness.
3. Build multiple alignments.
4. From the alignments, build gene trees.
5. Reconcile the gene tree to the species tree.

21.1.1 Phylogenetics

The two main pipelines for building trees are distance-based and character-based. Last lecture focused on distance-based pipeline. In distance-based, you form a pair-wise distance matrix using Jukes-Cantor or Kimura. Then use Neighbor Joining or UPGMA to reconstruct a tree from the matrix. Distance based pipelines use a fixed number of steps so and UPGMA and NJ have a running time is $O(n^3)$. However, there are flaws to this approach. Distance-based metrics are overly simplified and under measure the rate of mutation because a nucleotide that gets mutated back to its original form is counted as not mutated.

Today’s lecture focuses on the character-based pipeline, which is NP Hard so we have to resort to heuristics. The basic idea is we want to search through different trees and test each one. We start with an initial tree, then compute the probability/likelihood, then explore the tree space using methods such as nearest neighbor interchange (NNI), compute the score again, loop and then return the tree with the highest score as the answer. Using NNI, we can go to all trees in the tree space. The problem is that the tree space is very big (why it is NP hard).

![Figure 21.1: Heuristic tree search in character-based reconstruction](image)

For the scoring metric, we want to maximize

Using the Felsenstein peeling algorithm, we can efficiently compute $P(X|T, B)$ by building up a dynamic programming problem. We can first look at site evolution along a single branch, then build on that and
look at sequence evolution and then look at site evolution along an entire tree. Site evolution uses the Jukes-Cantor model and has the definition

\[ P(x_1, x_2, x_3 | T, t) = \sum x_5 \sum x_6 \sum x_7 P(x_1 | x_5, t_1) P(x_2 | x_5, t_1) P(x_3 | x_6, t_3) P(x_4 | x_6, t_4) \]

Assuming site independence does not always hold for example in RNA coding regions the sites is not independent due to RNA folding. To move to sequence evolution over an entire tree, we assume branch independence once we condition on the parent sequence.

From the equation in sequence evolution over an entire tree, we need both internal nodes and leaves \((x_1, \ldots, x_{2n-1})\) but only leaves \((x_1, \ldots, x_n)\) are given so we need to marginalize over unknowns \((x_{n+1}, \ldots, x_{2n-1})\).

Using a factorization trick:

\[ P(x_1, x_2, x_3 | T, t) = \sum x_5 \sum x_6 \sum x_7 P(x_1 | x_5, t_1) P(x_2 | x_5, t_1) P(x_3 | x_6, t_3) P(x_4 | x_6, t_4) \]

The Peeling algorithm builds a DP table. Each entry contains the probability of seeing the leaf data below node I, given that node I has base a at site j. The leaves of the table are initialized based on the observed sequence. Entries are populated in post-order traversal. The runtime of the Peeling algorithm is \(O(nmk^2)\).

The Peeling algorithm scores one tree and we need to use the search algorithm to search for more trees. The runtime is for one tree while the entire runtime depends on how many trees you want to look at.
21.2 Inferring Orthologs/Paralogs, Gene Duplication and Loss

There are two commonly used trees. The species tree uses morphological characters, fossil evidence, etc to create a tree of how species are related (leaves are species). Gene trees look at specific genes in different species (leaves are genes).

Reconciliation is an algorithm to figure out how the gene tree fits inside the species tree. It maps the vertices in the gene tree to vertices in the species tree.

We want to minimize the duplication/loss so we want to map events as low in the tree as possible to when they happened to minimize loss.

Duplication events map to the same as both of its children. Loss event maps to gap in the mapping. Gene tree accuracy is important; even one branch misplaced can dramatically increases error.

21.3 Learning Across Gene Trees

If we knew the species tree we could know beforehand that we expect the branch to be longer. We can develop a model for what kind of branch lengths we can expect. We can use conserved gene order to tell orthologs and build trees.
Figure 21.9: Maximum Parsimony Reconciliation Recursive Algorithm

Figure 21.10: Using species trees to improve gene tree reconstruction.

Figure 21.11: We can develop a model for what kind of branch lengths we can expect. We can use conserved gene order to tell orthologs and build trees.

When gene is fast evolving in one species, it is fast evolving in all species. We can model a branch length as two different rate components. One is gene specific (present across all species) and a species specific which is customized to a specific species. This method greatly improves reconstruction accuracy.

### 21.4 Modeling Population and Allele Frequencies

People keep sequencing genomes so looking at how populations evolve is becoming more and more important and feasible. The Wright-fisher model is used to study drifts, bottlenecks, etc. The coalescent model combines the Wright-fisher with trees. Wright-fisher was designed to study the effect of finite population sizes. We
Figure 21.12: Branch length can be modeled as two different rate components: gene specific and species specific.

Need to assume population size is fixed at $N$, random mating, non-overlapping generations.

Figure 21.13: The Wright-Fisher model

Continue for many generations and ignore ordering of chromosomes.

Figure 21.14: The Wright-Fisher model continued over many generations and ignoring the ordering of chromosomes.
The coalescent model only focuses on the genealogy. It only is concerned about the lineages we have sequences for; do not have to worry about others. It is a probabilistic model that works backwards in time to find when they have common ancestors.

Say we have 2N individuals, what is the probability that k lineages do not have any coalescent events in parental generation? What is the probability that the first coalescent of k lineages is at t generations? This process can be seen as a geometric distribution.

Can repeat to find when all individuals coalesce. Each branch of species tree can be seen as having its own Wright-Fisher inside of it.

21.5 SPIDIR: Background

As presented in the supplementary information for SPIDIR, a gene family is the set of genes that are descendents of a single gene in the most recent common ancestor (MRCA) of all species under consideration. Furthermore, genetic sequences undergo evolution at multiple scales, namely at the level of base pairs, and at the level of genes. In the context of this lecture, two genes are orthologs if their MRCA is a speciation event; two genes are paralogs if their MRCA is a duplication event.

In the genomic era, the species of a modern genes is often known; ancestral genes can be inferred by reconciling gene- and species-trees. A reconciliation maps every gene-tree node to a species-tree node. A common technique is to perform Maximum Parsimony Reconciliation (MPR), which finds the reconciliation
Figure 21.17: Multispecies Coalescent Model. Leaf branches track one lineage. There is a lag time from when population separated and when two actual gene lineages find a common ancestor. The rate of coalescent slows down as N gets bigger and for short branches. Deep coalescent is depicted in light blue for three lineages. The species and gene tree are incongruent since C and D are sisters in gene tree but not the species tree. There is a \( \frac{2}{3} \) chance that incongruence will occur because once we get to the light blue section, the Wright-fisher is memory less and there is only \( \frac{1}{3} \) chance that it will be congruent. Effect of incongruence is called incomplete lineage sorting.

R implying the fewest number of duplications or losses using the recursion over inner nodes \( v \) of a gene tree \( G \). MPR fist maps each leaf of the gene tree to the corresponding species leaf of the species tree. Then the internal nodes of \( G \) are mapped recursively:

\[
R(v) = MRCA(R(right(v)), R(left(v)))
\]

If a speciation event and its ancestral node are mapped to the same node on the species tree. Then the ancestral node must be an duplication event.

Using MPR, the accuracy of the gene tree is crucial. Suboptimal gene trees may lead to an excess of loss and duplication events. For example, if just one branch is misplaced (as in ??) then reconciliation infers 3 losses and 1 duplication event. In [??], the authors show that the contemporaneous current gene tree methods perform poorly (60% accuracy) on single genes. But if we have longer concatenated genes, then accuracy may go up towards 100%. Furthermore, very quickly or slowly evolving genes carry less information as compared with moderately diverging sequences (40-50% sequence identity), and perform correspondingly worse. As corroborated by simulations, single genes lack sufficient information to reproduce the correct species tree. Average genes are too short and contains too few phylogenetically informative characters. While many early gene tree construction algorithms ignored species information, algorithms like SPIDIR capitalize on the insight that the species tree can provide additional information which can be leveraged for gene tree construction. Synteny can be used to independently test the relative accuracy of different gene tree reconstructions. This is because syntenic blocks are regions of the genome where recently diverged organisms have the same gene order, and contain much more information than single genes.

Figure 21.18: MPR reconciliation of genes and species tree.

There have been a number of recent phylogenomic algorithms including: RIO [??], which uses neighbor
joining (NJ) and bootstrapping to deal with incogruencies, Orthostrapper [? ], which uses NJ and reconciles to a vague species tree, TreeFAM [? ], which uses human curation of gene trees as well as many others. A number of algorithms take a more similar track to SPIDIR [? ], including [? ], a probabilistic reconciliation algorithm [? ], a Bayesian method with a clock,[? ],and parsimony method using species tree , as well as more recent developments: [? ] a Bayesian method with relaxed clock and [? ], a Bayesian method with gene and species specific relaxed rates (an extension to SPIDIR).

21.6 SPIDIR: Method and Model

SPIDIR exemplifies an iterative algorithm for gene tree construction using the species tree. In SPIDIR, the authors define a generative model for gene-tree evolution. This consists of a prior for gene-tree topology and branch lengths. SPIDIR uses a birth and death process to model duplications and losses (which informs the prior on topology) and then then learns gene-specific and species-specific substitution rates (which inform the prior on branch lengths). SPIDIR is a Maximum a posteriori (MAP) method, and, as such, enjoys several nice optimality criteria.

In terms of the estimation problem, the full SPIDIR model appears as follows:

\[
\text{argmax}_{L, T, R | D, S, \Theta} (L, T, R | D, S, \Theta) = \text{argmax}_{L, T, R | D, S, \Theta} (D | T, L) P(L | T, R, S, \Theta) P(T, R | S, \Theta)
\]

The parameters in the above equation are:  
- \( D = \) alignment data 
- \( L = \) branch length 
- \( T = \) gene tree topology 
- \( R = \) reconciliation 
- \( S = \) species tree (expressed in times) 
- \( \Theta = (\) gene and species specific parameters [estimated using EM training], \( \lambda, \mu \) dup/loss parameters). This model can be understood through the three terms in the right hand expression, namely:

1. the sequence model– \( P(D | T, L) \). The authors used the common HKY model for sequence substitutions, which unifies Kimura’s two parameter model for transitions and transversions with Felsenstein’s model where substitution rate depends upon nucleotide equilibrium frequency.

2. the first prior term, for the rates model– \( P(L | T, R, S, \Theta) \), which the authors compute numerically after learning species and gene specific rates.

3. the second prior term, for the duplication/loss model– \( P(T, R | S, \Theta) \), which the authors describe using a birth and death process.

Having a rates model is very rates model very useful, since mutation rates are quite variable across genes. In the lecture, we saw how rates were well described by a decomposition into gene and species specific rates. In lecture we saw that an inverse gamma distribution appears to parametrize the gene specific substitution rates, and we were told that a gamma distribution apparently captures species specific substitution rates. Accounting for gene and species specific rates allows SPIDIR to build gene trees more accurately than previous methods. A training set for learning rate parameters can be chosen from gene trees which are congruent to the species tree. An important algorithmic concern for gene tree reconstructions is devising a fast tree search method. In lecture, we saw how the tree search could be sped up by only computing the full \( \text{argmax}_{L, T, R | D, S, \Theta} (L, T, R | D, S, \Theta) \) for trees with high prior probabilities. This is accomplished through a
computational pipeline where in each iteration 100s of trees are proposed by some heuristic. The topology prior \( P(T, R|D, S, \Theta) \) can be computed quickly. This is used as a filter where only the topologies with high prior probabilities are selected as candidates for the full likelihood computation.

The performance of SPIDIR was tested on a real dataset of 21 fungi. SPIDER recovered over 96% of the synteny orthologs while other algorithms found less than 65%. As a result, SPIDER invoked much fewer number of duplications and losses.

### 21.7 Conclusion

Incorporating species tree information into the gene tree building process via introducing separate gene and species substitution rates allows for accurate parsimonious gene tree reconstructions. Previous gene tree reconstructions probably vastly overestimated the number of duplication and loss events. Reconstructing gene trees for large families remains a challenging problem.

### 21.8 Current Research Directions

### 21.9 Further Reading

### 21.10 Tools and Techniques

### 21.11 What Have We Learned?

### Bibliography


List of figures

Allele frequency comparison between populations ??
Projecting populations onto principal component of other populations ??
An admixture that fits Indian history. ??
22.1 Introduction

Humans share 99.9% of the same genetic information, and are 99% similar to chimpanzees. Learning about the 0.1% difference between humans can be used to understand population history, trace lineages, predict disease, and analyze natural selection trends. In this lecture, Dr. Reich explained how we can use this data to see evidence of gene flow between neanderthals and modern humans of Western Eurasian decent.

Last year, he examined India as a case of how genetic data can inform population history, which is included as an appendix.

22.2 Quick Survey of Human Genetic Variation

In the human genome, there is generally a polymorphism every 1000 bases, though there are regions of the genome where this rate can quadruple. These polymorphisms are markers of genetic variation. It is necessary to understand how genetic variation arises before attempting to analyze it. Single Nucleotide Polymorphisms (SNPs) are one manifestation of genetic variation. When SNPs occur, they are segregated according to recombination rates, advantages or disadvantages of the mutation, and the population structure that exists and continues during the lifespan of the SNP. Through the passing of generations, recombination splits the SNP haplotype into smaller blocks. The length of these blocks, then, is dependent on the rate of recombination and the stability of the recombination product. Therefore, the length of conserved haplotypes can be used to infer the age of mutation or its selection. An important consideration, though, is that the rate of recombination is not uniform across the genome; rather, there are recombination hot spots that can skew the measure of haplotype age or selectivity. This makes the haplotype blocks longer than expected under a uniform model.

To Dr. Reich, every place in the genome is best thought of as a tree when compared across individuals. But, depending on where are you look within the genome, this particular tree will be different than another particular tree you may get from a specific set of SNPs. The trick is to use the data that we have available on SNPs to infer the underlying trees, and then the overarching phylogenetic relationships. Take, for instance, the Y chromosome. It undergoes little to no recombination and thus can produce a high accuracy tree as it passed down from father to son. Likewise, we can take mitochondrial DNA, passed down from mother to child. While these trees can have high accuracy, other autosomal trees are confounded with recombination, and thus show lower accuracy to predict phylogenetic relationships. Gene trees are best made by looking at areas of low recombination, as recombination mixes trees. In general, there are about 1 to 2 recombinations per generation.

Humans show about 10,000 base-pairs of linkage, as we go back about 10,000 generations. Fruit fly linkage equilibrium blocks, on the other hand, are only a few hundred bases. Fixation will occur over time, proportional to the size of the population. For a population of about 10,000 it will take about 10,000 years to reach that point. When a population expands, genetic drift goes down. So, curiously enough, the variation in humans looks like what would have been formed in a population size of 10,000.

If long haplotypes are mapped to genetic trees, approximately half of the depth is on the first branch; most morphology changes are deep in the tree because there was more time to mutate. One simple model of mutation without natural selection is the Wright-Fisher neutral model which utilizes binomial sampling. In this model, a population will reach fixation (frequency 1), will die out (frequency 0), or continue to segregate. In the human genome, there are 10-20 million common SNPs. This is less diversity than chimpanzees, implying that humans are genetically closer to one another.

With this genetic similarity in mind, comparing human sub-populations can give information about common ancestors and suggest historical events. The similarity between two sub-populations can be measured by comparing allele frequencies in a scatter plot. If we plot the frequencies of SNPs across different populations on a scatterplot, we see more spread between more distant populations. The plot below, for example, shows the relative dissimilarity of European American and American Indian populations along with the greater similarity of European American and Chinese populations. The plots indicate that there was a divergence in the past between Chinese and Native Americans, evidence for the North American migration bottleneck that has been hypothesized by archaeologists. The spread among different populations within Africa is quite large. We can measure spread by Fst (which describes the variance).
Several current studies have shown that unsupervised clustering of genetic data can recover self-selected labels of ethnic identity. In Rosenberg’s experiment, a Bayesian clustering algorithm was developed. They took a sample size of 1000 people (50 populations, 20 people per population), and clustered those people by their SNP genetic data but they didn’t tag any of the people by populations, so they could see how the algorithm would cluster them without knowledge of ethnicity. They didn’t know what the optimal number of clusters was, so they tried 2, then 3, then 4, then 5, et cetera. What they found was that with 2 clusters, East-Asians and non-East-Asians were separated. With 3 clusters, Africans were separated from everyone else. With 4, East-Asians and Native Americans were separated. And then with 5, the smaller sub-populations began to emerge.

When waves of humans left Africa, genetic diversity decreased; the small numbers of people in the groups that left Africa allowed for serial founder events to occur. These serial founder events lead to the formation of sub-populations with less genetic diversity. This is evidenced by the fact that genetic diversity decreases moving out of Africa. West Africans have the highest diversity of any human sub-population.

22.3 Neanderthals and Modern Human Gene Flow

Recently, Dr. Reich worked with the Max Planck Institute as a population geneticist studying Neanderthal genetic data. He discussed with us the background of his research as part of the Neanderthal genome project, the draft sequence that they assembled, and then the evidence that’s been compiled for gene flow between modern humans and Neanderthals.

22.3.1 Background

Clear fossils from Neanderthals from 200,000 years ago exist in West Eurasia (Europe and Western Asia), which is far earlier than Homo erectus. The earliest fossils of us come from Ethiopia dating about 200,000 years ago. However, there is evidence that neanderthals and humans overlapped in time and space between 135,000 and 35,000 years ago.

The first place of contact could have occurred in The Levant, in Israel. There are human fossils from 120,000 years ago, then a gap, neanderthal fossils about 80,000 years ago, another gap, and then human fossils again 60,000 years ago. This is proof of an overlap in place, but not in time. In the upper paleolithic era, there was an explosion of populations out of Africa (the migration about 60,000 to 45,000 years ago). In Europe after 45,000 years ago, there are sites where neanderthals and humans exist in the fossil record side by side. (Eastern Eurasia is not well documented, so we have little evidence from there.) Since there is evidence that they co-existed, was there interbreeding? This is a question we wish to answer by examining population genomics.

Let’s take a look at how you go about finding and sequencing DNA from ancient remains. First, you have to obtain a bone sample with DNA from a neanderthal. Human DNA and Neanderthal DNA is very
similar (we are more similar to them than we are to chimps), so when sequencing short reads with very old DNA, its impossible to tell if the DNA is neanderthal or human. So, the cave is first classified as human or non-human, which helps to predict the origin of the bones. In sites of findings, typically lots of trash is left behind which is used to help classify the site (stone tools, particular technologies, the way meat was cut off animals, other trash). These caves are made up of lots of trash, and only the occasional bone. Even if you have a bone, it is still very unlikely that you have any salvageable DNA. In fact, 99% of the sequence of Neanderthals comes from only three long bones found in one site: the Vindija cave in Croatia (5.3 Gb, 1.3x full coverage). The paleontologists chose to sacrifice the long bones because they were less morphologically helpful.

Next, the DNA is sent to an ancient-DNA lab. Since they are 40,000 year old bones, there is very little DNA left in them. So, they are first screened for DNA. If they have it, is it primate DNA? Usually it is DNA from microbes and fungi that live in soil and digest us when we die. If it is primate DNA, is it contamination from the human (archeologist or lab tech) handling it? The difference between human and neanderthal DNA is 1/600 bp. The size of reads from a 40,000 year old bone sample is 30-40 bp. The reads are almost always identical for a human and neanderthal, so it is difficult to distinguish them.

Only about 1-10% of the DNA on old bones is the primates DNA. 89 DNA extracts were screened for neanderthals, but only 6 bones were actually sequenced (requires lack of contamination and high enough amount of DNA). The process of retrieving the DNA requires drilling beneath the bone surface (to minimize contamination) and taking samples from within. For the three long bones, less than 1 gram of bone powder was able to be obtained. Then the DNA is sequenced and aligned to a reference chimp genome. It is mapped to a chimp instead of a particular human because mapping to a human might cause bias if you are looking to see how the sequence relates to specific human sub-populations.

Most successful finds have been in cool limestone caves, where it is dry and cold and perhaps a bit basic. The best chance of preservation occurs in permafrost areas. Very little DNA is recoverable from the tropics. The tropics have a great fossil record, but DNA is much harder to obtain. Since most bones dont yield enough or good DNA, scientists have the screen samples over and over again until they eventually find a good one.

22.3.2 Draft Sequence

The neanderthal DNA had short reads, about 37 bp on average. There are lots of holes due to mutations caused by time eroding the DNA. However, there some characteristic mutations that occur on DNA thats been sitting for very long periods of time. There is a tendency to see C to T mutations, and G to A mutations. Over time, a methyl group gets knocked off of a C, which causes it to resemble to U. When PCR is used to amplify the DNA for sequencing, the polymerase sees a U and repairs it to a T. The G to A mutations are just the result of seeing that on the opposite strand, so really the important mutation to worry about is the C to T. This mutation is seen about 2% of the time! In order to combat this, scientists use a special enzyme now that recognizes the U, and instead of replacing it with a T, simply cuts the strand where it sees the mutation. This helps to identify those sites.

The average fragment size is quite small, and the error rate is still 0.1% - 0.3%. One way to combat the mutations is the note that on a double stranded fragment, the DNA is frayed towards the ends, where it becomes single stranded for about 10 bp. There tend to be high rates of mutations in the first and last 10 bases, but high quality elsewhere (so C to T mutations in the beginning and G to A in the end). In chimps, the most common mutations are transitions (purine to purine, pyrimidine to pyrimidine), and transversions are much rarer. The same goes for humans. Since the G to A and C to T mutations are transitions, it can be determined that there are about 4x more mutations in the Neanderthal DNA than if it were fresh by noting the number of transitions seen compared to the number of transversions seen (by comparing Neanderthal to human DNA). Transversions have a fairly stable rate of occurrence, so that ratio helps determine how much error has occurred through C to T mutations.

We are now able to get human contamination of artifact DNA down to around 1%. When the DNA is brought in, as soon as it is removed from the bone it is bar coded with a 4 bp tag (originally it was 4, now it is 7). That allows you to avoid contamination at any later point in the experiment, but not earlier. Extraction is also done in a clean room with UV light, after having washed the bone. Mitochondrial DNA is helpful for distinguishing what percent of the sample is contaminated with human DNA. Mitochondrial DNA is filled
with characteristic event sites in different species - all Neanderthals were of one type, all humans of another (called reciprocally monophylogenetic). The contamination can be measured by counting the ratio of those sites. In the Neanderthal DNA, contamination was present, but it was \( \frac{1}{300} \) to 11000. So we can, in fact, still get lots of usable data from this after all.

After aligning the chimp, Neanderthal, and modern human sequences, we can measure the distance that Neanderthals are, on the scale from humans to chimps. This distance is only about 12.7% from the human reference sequence. A French sample measures about 8% distance from the reference sequence, and a Bushman about 10.3%. What this says is that the Neanderthal DNA is within our range of variation as a species.

### 22.3.3 Evidence for Gene Flow

1. First, let's look at a comparison test. Take two randomly chosen populations, sequence both, and for each different SNP, check to see which population the Neanderthal DNA matched. This was done for 8 sequences. When Eurasians were compared with Eurasians, there was little difference. When Africans were compared with Africans, there was also little difference. However, when Africans were compared with non-Africans, Neanderthal SNPs much more highly matched the non-African DNA. This is evidence that there was mating and gene flow between Neanderthals and Eurasian modern humans.

2. Second, we'll take a look at a long range haplotype study done at Berkeley. These researchers picked long range sections of the genome and compared them among randomly chosen humans from various populations. When you look to see where the deepest branch of the tree constructed from that haplotype is, it almost always comes from an African population. However, occasionally non-Africans have the deepest branch. The study found that there were 12 regions where non-Africans have the deepest branch. When this data was used to analyze the Neanderthal genome, it was found that \( \frac{10}{12} \) of these regions in non-Africans matched Neanderthals more than the matched the human reference sequence (a compilation of sequences from various populations). This is evidence of that haplotype actually being of Neanderthal origin.

3. Lastly, there is a bigger divergence than expect among humans. The average split between a Neanderthal and a human is about 800,000 years. The typical divergence between two humans is about 500,000 years. When looking at african and non-african sequences, regions of low divergence emerged in non-african sequences when compared with Neanderthal material. The regions found were highly enriched for Neanderthal material (94% Neanderthal), which would increase the average divergence between humans (as the standard Neanderthal - human divergence is about 800,000 years).

### 22.4 Discussion

There was an example of a 50,000 year old bone found in southern Siberia, where the mtDNA was sequenced, that appears to be an out-group to both Neanderthals and modern humans. It was a little finger bone of a child. It is twice as deep in the phylogenetic tree as either of them, and has 1.9x coverage. These pieces of the ancestral DNA puzzle help us piece together human history and before. They serve to help us understand where we came from.

The bottleneck caused by the migration from Africa is only one example of many that have occurred. Most scientists usually concentrate on the age and intensity of events and not necessarily the duration, but the duration is very important because long bottlenecks create a smaller range of diversity. One way to help tell the length of a bottleneck is to determine if any new variations arose during it, as that occurs during longer bottlenecks, and as they will help distinguish how long it lasted. That change in range of diversity
is also what helped create the different human sub-populations that became geographically isolated. This is just another way that population genomics can be useful for helping to piece together information.

Today, Dr. Reich showed how genetic differences between species (specifically here within primates) can be used to help understand the phylogenetic tree from which we are all derived. We looked at the case study of comparisons with Neanderthal DNA, learned about how ancient DNA samples are obtained, how sequences are found and interpreted, and how that evidence shows high likelihood of interbreeding between modern humans (of eurasian descent) and Neanderthals. Those very small differences between one species and the next, and within species, are what allow us to deduce a great deal of this history through population genetics.

### 22.5 Current Research Directions

### 22.6 Further Reading

#### 22.6.1 Fall 2009 Discussion Topic: Genomic Variation in 25 Groups in India

There is a general taxonomy for studying population relationships with genetic data. The first general type of study utilizes both phylogeny and migration data. It fits the phylogenies to Fst values, values of sub-population heterozygosity (pioneered by Cavalli-Sforza and Edwards in 2267). TODO: cite the paper where this is discussed in more detail. This method also makes use of synthetic maps and Principal Components Analysis. The primary downside to analyzing population data this way is uncertainty about results. There are mathematical and edge effects in the data processing that cannot be predicted. Also, certain groups have shown that separate, bounded mixing populations can produce significant-seeming principal components by chance. Even if the results of the study are correct, then, they are also uncertain.

The second method of analyzing sub-population relationships is genetic clustering. Clusters can be formed using self-defined ancestry or the STRUCTURE database. This method is overused and can over-fit the data; the composition of the database can bias the clustering results.

Technological advances and increased data collection, though, have produced data sets that are 10,000 times larger than before, meaning that most specific claims can be disproved by some subset of data. So in effect, many models that are predicted either by phylogeny and migration or genetic clustering will be disproved at some point, leading to large-scale confusion of results. One solution to this problem is to use a simple model that makes a statement that is both useful and has less probability of being falsified.

Past surveys in India have studied such aspects as anthropometric variation, mtDNA, and the Y chromosome. The anthropometric study looked at significant differences in physical characteristics between groups separated by geography and ethnicity. The results showed variation much higher than that of Europe. The mtDNA study was a survey of maternal lineage and the results suggested that there was a single Indian tree such that age of lineage could be inferred by the number of mutations. The data also showed that Indian populations were separated from non-Indian populations at least 40,000 years ago. Finally, the Y chromosome study looked at paternal lineage and showed a more recent similarity to Middle Eastern men and dependencies on geography and caste. This data conflicts with the mtDNA results. One possible explanation is that there was a more recent male migration. Either way, the genetic studies done in India have served to show its genetic complexity. The high genetic variation, dissimilarity with other samples, and difficulty of obtaining more samples lead to India being left out of HapMap, the 1000 Genomes Project, and the HGDP.

For David Reich and collaborators study of India, 25 Indian groups were chosen to represent various geographies, language roots, and ethnicities. The raw data included five samples for each of the twenty five groups. Even though this number seems small, the number of SNPs from each sample has a lot of information. Approximately five hundred thousand markers were genotyped per individual. Looking at the data to emerge from the study, if Principal Components Analysis is used on data from West Eurasians and Asians, and if the Indian populations are compared using the same components, the India Cline emerges. This shows a gradient of similarity that might indicate a staggered divergence of Indian populations and European populations.
22.6.2 Almost All Mainland Indian Groups are Mixed

Further analysis of the India Cline phenomenon produces interesting results. For instance, some Pakistani sub-populations have ancestry that also falls along the Indian Cline. Populations can be projected onto the principal components of other populations: South Asians projected onto Chinese and European principal components produces a linear effect (the India Cline), while Europeans projected onto South Asian and Chinese principal components does not. One interpretation is that Indian ancestry shows more variability than the other groups. A similar variability assessment appears when comparing African to non-African populations. Two tree hypotheses emerge from this analysis:

1. there were serial founder events in India’s history or
2. there was gene flow between ancestral populations.

The authors developed a formal four population test to test ancestry hypotheses in the presence of admixture or other confounding effects. The test takes a proposed tree topology and sums over all SNPs of \((P_{p1} \times P_{p2})(P_{p3} \times P_{p4})\), where \(P\) values are frequencies for the four populations. If the proposed tree is correct, the correlation will be 0 and the populations in question form a clade. This method is resistant to several problems that limit other models. A complete model can be built to fit history. The topology information from the admixture graphs can be augmented with Fst values through a fitting procedure. This method makes no assumptions about population split times, expansion and contractions, and duration of gene flow, resulting in a more robust estimation procedure.

![Figure 22.2: Populations can be projected onto the principal components of other populations: South Asians projected onto Chinese and European principal components produces a linear effect (the India Cline), while Europeans projected onto South Asian and Chinese principal components does not.](image)

Furthermore, estimating the mixture proportions using the 4 population statistic gives error estimates for each of the groups on the tree. Complicated history does not factor into this calculation, as long as the topology as determined by the 4-population test is valid.

These tests and the cline analysis allowed the authors to determine the relative strength of Ancestral North Indian and Ancestral South Indian ancestry in each representative population sample. They found that high Ancestral North Indian ancestry is correlated with traditionally higher caste and certain language groupings. Furthermore, Ancestral North Indian (ANI) and South Indian (ASI) ancestry is as different from Chinese as European.

22.6.3 Population structure in India is different from Europe

Population structure in India is much less correlated with geography than in Europe. Even correcting populations for language, geographic, and social status differences, the Fst value is 0.007, about 7 times that
of the most divergent populations in Europe. An open question is whether this could be due to missing (largely India-specific) SNPs on the genotyping arrays. This is because the set of targeted SNPs were identified primarily from the HapMap project, which did not include Indian sources.

Most Indian genetic variation does not arise from events outside India. Additionally, consanguineous marriages cannot explain the signal. Many serial founder events, perhaps tied to the castes or precursor groups, could contribute. Analyzing a single group at a time, it becomes apparent that castes and subcastes have a lot of endogamy. The autocorrelation of allele sharing between pairs of samples within a group is used to determine whether a founder event occurred and its relative age. There are segments of DNA from a founder, many indicating events more than 1000 years old. In most groups there is evidence for a strong, ancient founder event and subsequent endogamy. This stands in contrast to the population structure in most of Europe or Africa, where more population mixing occurs (less endogamy).

These serial founder events and their resulting structure have important medical implications. The strong founder events followed by endogamy and some mixing have lead to groups that have strong propensities for various recessive diseases. This structure means that Indian groups have a collection of prevalent diseases, similar to those already known in other groups, such as Ashkenazi Jews or Finns. Unique variation within India means that linkages to disease alleles prevalent in India might not be discoverable using only non-Indian data sources. A small number of samples are needed from each group, and more groups, to better map these recessive diseases. These maps can then be used to better predict disease patterns in India.

22.6.4 Discussion

Overall, strong founder events followed by endogamy have given India more substructure than Europe. All surveyed tribal and caste groups show a strong mixing of ANI and ASI ancestry, varying between 35% and 75% ANI identity. Estimating the time and mechanism of the ANI-ASI mixture is currently a high priority. Additionally, future studies will determine whether and how new techniques like the 4-population test and admixture graphs can be applied to other populations.
22.7 Tools and Techniques

22.8 What Have We Learned?

Bibliography


CHAPTER

TWENTYTHREE

POPULATION GENETIC VARIATION

Guest Lecture by Pardis Sabeti

List of figures

Changes in Allele Frequencies Over Time
Isolated Populations
Genomic Signals of Natural Selection

243
23.1 Introduction

The ability to sequence find patterns in genomes is a fundamental tool in research into the evolution of organisms, especially Humans. There are various algorithms to find the similarities between the genome of our species and another, but how do we find the divergences and how long ago did they occur? In other words, how do we measure selection in the Human Genome? This lecture aims to outline the main criteria and features that are analyzed to answer these questions.

23.2 Polymorphisms

Polymorphisms are differences in appearance (Phenotype) amongst members of the same species. Since all polymorphisms have a genetic basis, they can be characterized into types based on the mutation in the genome.

Single Nucleotide Polymorphisms (SNPs)  • The mutation of only a single nucleotide base within a sequence. In most cases, this type of change has no consequence. However, there are some cases where this causes a major change.
  • Prime example is when glutamic acid (GAG) valine (GTG) in hemoglobin and causes Sickle Cell Anemia

Variable Number Tandem Repeats  • When the copying machinery in the body is transcribing repeats within the genome, but loses track of how many repeats its made and makes more repeats then there originally was.
  • Prime example is a triple CAG repeat which causes Huntingtons disease, where there is gradual muscle control loss and severe neurological degradation. Also an example of the phenomenon of anticipation where if it is inherited by the next generation, it becomes more severe in that generation

Insertion/Deletion  • When certain nucleotide bases are just forgotten to be transcribed or extra nucleotides are transcribed into a sequence.
  • Worst when only 1 or 2 are deleted or added since this will shift the frame of reading codons (groups of 3 nucleotides) for instructions
  • Prime example is deletions in the CFTR gene, which codes for chloride channels in the lungs and may cause Cystic Fibrosis where the patient cannot clear mucous in the lungs and causes infection

23.2.1 Allele and Genotype Frequencies

Hardy Weinberg Principle: Allele and Genotype frequencies within a population will remain at constant equilibrium unless there is an outside influence and/or interaction.

Assumptions in Hardy-Weinberg:

• All mating in the population is at random, there is no inbreeding or arranged pairings
• There is no random mutations in the population
• No migration of the species or introduction of another subpopulation into the general population; isolated
• No natural selection, all individuals have equal probability of survival
• The population observed is very large
• Allele frequency drives future genotype frequency (Prevalent allele drives Prevalent genotype)
In a Hardy Weinberg Equilibrium, for two genotypes A and T, occurring with probability $p$ and $q = 1-p$, respectively, the probabilities of finding the homozygous AA or TT (pp or qq, respectively) or heterozygous (2pq) genotypes can be described by the equation:

$$p^2 + 2pq + q^2 = 1$$

This equation gives a table of probabilities for each genotype, which can be compared with the observed genotype frequencies using statistical error tests such as the chi-squared test to determine if the Hardy-Weinberg model is applicable.

In a small population that violates one criteria of the Hardy-Weinberg, when genetic drift (a mutation) occurs, it will always either disappear (frequency = 0) from the population or become prevalent in a species - this is called fixation; in general, 99% of mutations disappear. Shown below is a simulation of a mutations prevalence in a finite-sized population over time: both perform random walks, with one mutation disappearing and the other becoming prevalent:

![Figure 23.1: Changes in allele frequency over time](image)

Once a mutation has disappeared, the only way to have it reappear is the introduction of a new mutation into the species population. For humans, it is believed that a given mutation, evolving neutrally, should fixate to 0 or 1 (within, e.g., 5%) within a few million years. However, under selection this will happen much faster.

### 23.2.2 Ancestral State of Polymorphisms

How do we determine for e.g., a SNP, which allele was the original and which is the mutation? Traces of the ancestral state can be done by comparing the genomes of a species and a closely related species, or outgroup (e.g. humans and chimpanzees) with a known phylogenetic tree. Mutations can occur anywhere along this tree sometimes mutations at the split fix differently in different populations (fixed difference), in which the entire populations differ in genotype. However, recently occurring mutations have not fixed, and a polymorphism will be present in one species but fully absent in the other (simultaneous mutations in both species are very rare). In this case, the derived variant is the version of the polymorphism appearing after the split, while the ancestral variant is the version shared by both species. There is one caveat perhaps 1% of the time, something more complex (e.g. double mutations) may occur, and this simple model does not hold.

### 23.2.3 Measuring Derived Allele Frequencies

The derived allele frequency (i.e., the frequency of the derived variant in the population) can be very easily measured. In doing this there is an implicit assumption that the population is homogeneous. However, in fact there could be some unseen divide between two groups that cause them to evolve separately (shown below):

In this case, the prevalence of the variants among subpopulations is different, and Hardy-Weinberg is violated.

One way to quantify this is to use the Fixation Index (Fst) to compare subpopulations within a species. In reality, it is merely a proportion of the total heterozygosity found in a species in a given subpopulation Fst estimates reduction in heterozygosity (2pq with alleles p and q) expected when 2 different populations...
are erroneously grouped given that there is in a population \( n \) alleles with frequencies \( p_i \) where \( 1 \leq i \leq n \) and homozygosity, \( G \), of the population is given by:

\[
\sum_{i=1}^{n} p_i^2
\]

And the total heterozygosity in the population is given by \( 1 - G \)

\[
F_{st} = \frac{\text{Heterozygosity(total)} - \text{Heterozygosity(subpopulation)}}{\text{Heterozygosity(total)}}
\]

In the case of the figure above, there is no heterozygosity between the populations, so \( F_{st} = 1 \). In reality within one species (ex: humans) the \( F_{st} \) will be small (0.0625). For actual tests, \( F_{st} \) is computed either by clustering sub-populations randomly, or using some simple/obvious characteristic, e.g., ethnicity.

### 23.3 Genetic Linkage

The second law in Mendelian inheritance is the law of independent assortment:

\begin{quote}
\textit{Law of Independent Assortment: Alleles of different genes assort independently without influence from other alleles.}
\end{quote}

When this law holds, there is no correlation between different polymorphisms. Then the probability of a haplotype (a given set of polymorphisms) is simply the product of the individual polymorphism probabilities.

Of course, at first sight this law seems impossible: genes lie on chromosomes, and chromosomes are passed on as units to offspring. Therefore, two polymorphisms occurring on the same chromosome should be perfectly correlated (perfectly linked). However, genetic recombination events, in which segments of DNA on homologous chromosomes are swapped, will over time reduce the correlation between polymorphisms. Over a suitably long time interval, recombination will remove the linkage between two polymorphisms; they are said then to be in equilibrium. When, on the other hand, the polymorphisms are correlated (either because they are both very recent, or as we will see below, by positive selection), we have Linkage Disequilibrium (LD). The amount of disequilibrium is the difference between the observed haplotype frequencies and those predicted in equilibrium.

We can use this LD measurement (\( = D \)) to find the difference between observed and expected assortments. If there are two Alleles (1 and 2) and two loci (A and B) we can calculate the haplotype probabilities and find expected allele frequency probabilities and compare to observed probabilities

- **Haplotype frequencies**
  - \( P(A_1) = x_{11} \)
  - \( P(B_1) = x_{12} \)
  - \( P(A_2) = x_{21} \)
  - \( P(B_2) = x_{22} \)

- **Allele Frequency**
  - \( P_{11} = x_{11} + x_{12} \)
\[ P_{21} = x_{21} + x_{22} \]
\[ P_{12} = x_{11} + x_{21} \]
\[ P_{22} = x_{12} + x_{22} \]

- \( D = P_{11} * P_{22} * P_{12} * P_{21} \)

Using the value \( D_{\text{max}}\), the maximum value of \( D \) with given allele frequencies, it can compared to \( D \) in the relationship:
\[ D' = D \frac{D_{\text{max}}}{D} \]

Where \( D' \) is equal the maximum linkage disequilibrium or complete skew for the given alleles and allele frequencies. \( D_{\text{max}} \) can be found by taking the smaller of the expected haplotype frequencies \( P(A1B2) \) or \( P(A2B1) \). If there is full combination and complete independent sorting, then it is in complete equilibrium in which case \( D' = 0 \) for completely unlinked. Vice versa, a value of \( D = 1 \) will demonstrate total linkage.

The key point for this lecture is that mutations that have occurred relatively recently have not had time to be broken down by Linkage Disequilibrium. Normally, such a mutation will not be very common. However, if it is under positive selection, the mutation will be much more prevalent in the population than expected. Therefore, by carefully combining a measure of LD and derived allele frequency we can determine of a region is under positive selection.

Decay of Linkage Disequilibrium is driven by recombination rate and time (in generations) and has an exponential decay. For a higher recombination rate, linkage disequilibrium will decay faster in a shorter amount of time. The hard part in the test for natural selection is actually getting the background recombination rate, as it is difficult to estimate and depends on the location in the genome. Comparison of genomic data across multiple species helps a lot in determining these background rates.

### 23.3.1 Correlation Coefficient \( r^2 \)

Answers how predictive is allele at locus A of allele at locus B

\[ r^2 = \frac{D^2}{P(A1)P(A2)P(B1)P(B2)} \]

As the value of \( r^2 \) approaches 1, the more it can be said that two alleles at two loci can be correlated.

There may also be Linkage Disequilibrium between two haplotypes, but the haplotypes are not correlated at all. The correlation coefficient is particularly interesting when studying disease mapping where information at locus A does not predict a disease where as locus B does. Or there could be the case where neither locus A nor locus B predict disease but loci AB where A1B2 does predict a particular disease. With many linkages however, we can hope to generalize and tag many haplotypes.

### 23.4 Natural Selection

In the mid 2300s there were many papers already on the theory of evolution; however, Darwin and Wallace provide mechanisms of natural selection that lead to evolution. Not until 70 years later (1948) did we have an example of how this could occur in humans: J.B.S Haldanes Malaria Hypothesis showed a correlation between genetic mutations in red blood cells and the distribution of malaria prevalence. It turned out that individuals with these mutations (e.g., sickle cell mutations) had a resistance to malaria. Therefore, there is a direct environmental pressure for genetic mutation.

Another example is LCT, lactose tolerance (lasting into adulthood). However, these explicit examples were very hard to construct, since the investigators did not have genetic data. Now, in hindsight, we can search the genome for regions with the same patterns as these known examples to identify regions undergoing natural selection. This begs the question, what are these signals?

### 23.4.1 Genomics Signals of Natural Selection

- Exponential prevalence of a feature in sequential generations
- Mutation that helps species prosper
Tests

- **Long range correlations (iHs, Xp, EHH):** If we tag genetic sequences on an allele and try to reconstruct, we end up with a broken haplotype and the number of breaks or color changes is directly correlated to the number of recombinations and to how old or long ago it occurred.

![Genomic Signals of Natural Selection](image)

- **SWEEP** program developed by Pardis Sabeti, Ben Fry and Patrick Varilly. SWEEP detects evidence of natural selection by analyzing haplotype structures in the genome and using Long Range Haplotype (LRH) test. It looks for high frequency alleles with long range Linkage Disequilibrium. This suggests that there was a large scale proliferation of a haplotype that occurred at a rate greater than recombination could break it from its markers.

- **High Frequency derived** Look for large spikes in the frequency of derived alleles in set positions.

- **High Differentiation (Fst)** Large spikes in differentiation at certain positions.

Using these tests, we can investigate selected regions. One problem is that, while a single SNP may be under positive selection and proliferate, nearby SNPs will hitchhike along. It is difficult to distinguish the SNP under selection from the hitchhikers with only one test. Under selection, these tests are strongly correlated; however, in the absence of selection they are generally independent. Therefore, by employing a composite statistic built from all of these tests, it is possible to isolate out the individual SNP under selection.

Examples where a single SNP has been implicated in a trait:

- Chr15 Skin pigmentation in Northern Europe
- Chr2 Hair traits in Asia
- Chr10 Unknown trait in Asia
- Chr12 Unknown Trait in Africa

The International HapMap Project aims to catalog the genomes of humans from various countries and regions and find similarities and differences to help researchers find genes that will benefit the advance in disease treatment and administration of health related technologies.

### 23.5 Current Research Directions

### 23.6 Further Reading

- Application to new data sets as they become available
• Additional genotyping and sequencing
• Functional validation
• Design of model organisms

23.7 Tools and Techniques

23.8 What Have We Learned?

Bibliography
CHAPTER TWENTYFOUR

MEDICAL GENETICS – THE PAST TO THE PRESENT

Guest Lecture by
Mark J. Daly (PhD)
Scribed by Anna Ayuso

List of figures

Disease and Quantitative Traits Have Genetic Components ??
Drug Development Cycle ??
Sources of Genetic Variation ??
Redundancy in Genetic Data ??
Single Gene Diseases ??
Discovery of Genes for Different Disease Types vs. Time ??
Example Linkage Analysis Pedigree ??
Inheritance Vector Example ??
Manhattan and Q-Q Plots from GWAS ??
24.1 Introduction

Mark J. Daly PhD, an Associate Professor at the Massachusetts General Hospital/Harvard Medical School and an Associate Member of the Broad Institute, presented on the field of Medical Genetics. The lecture explains how the nature of genetics allows for causal pathways to be inferred about various diseases by applying statistical and computational methods to Genomic data. Linkage Analysis and Genome Wide Association studies are two methods used for the purpose of understanding these disease pathways better and both methods are described in greater detail. The complexities of human disease are also laid out to aid in understanding the difficulties of this prediction problem. One final key point of this lecture is that rapidly changing technology has provided an abundance of novel data and this has enabled the application of new methods and the discovery of new insights into Human Genetics.

24.2 Genetics vs. Environment

The Genetics vs. Environment debate used to run more rampant before the recent explosion of data and studies have confirmed the contribution of each factor to a variety of different diseases and traits. It is generally agreed upon today that both genetics and environment play a factor in the manifestation of diseases and traits at differing levels of contribution. There are many diseases and traits that have strongly identified genetic components. This observation is what has fueled the emphasis on applying computational methods to finding genes responsible for diseases. One example of a disease that has a strong genetic basis is type 2 diabetes. Twin and sibling studies have enabled these findings. If you have type 2 diabetes, studies have shown that the risk to your neighbor is 5-10%, to your sibling is 30%, and to your identical twin is 80-90%. Studies like this have demonstrated that inherited variation in DNA sequence affects disease risk.

![Diseases and Quantitative Traits](image)

Figure 24.1: Many diseases and quantitative traits have genetic components.

24.3 Goals of Studying Human Genetics

Aside from the sheer curiosity of understanding ourselves better, there are several goals in understanding the causal pathways of disease by implicating genes of interest:

- Directing downstream research in disease
- Enabling rational drug development
- Potential for improved diagnostics

? depicts the cycle of drug development. First, researchers hypothesize a possible target of interest that might be related to a disease. The cycle is carried out of evaluating the biochemistry of this target, testing the target in animal models, and then finally performing clinical trials. The vast majority of targets that
are chosen for drugs end up not making it to the end of this process due to lack of efficacy in clinical trials. Since many of these targets are guesses, this yields a less than fruitful process. Studying Human Genetics with computational approaches could potentially allow drug researchers to front load the process and achieve a higher success rate in identifying targets that will yield positive results in treating a disease. This front loaded process is how Statins came to be.

Statins are a class of drugs used to lower cholesterol by inhibiting an enzyme that produces cholesterol in the liver. Dr. Michael Brown and Dr. Joseph Goldstein won the Nobel Prize in Physiology or Medicine in 1985 for discovering the underlying mechanism of cholesterol metabolism that led to the development of Statins. They were able to isolate the cause of extreme familial hypercholesterolemia to a mutation that removed receptors whose purpose it was to remove cholesterol from the blood. Understanding the precise point of failure allowed for an effective drug therapy that targeted the malfunction.

Figure 24.2: Drug Development Cycle

24.4 Population Variance

Understanding the way in which people differ genetically is important to potentially finding a culprit gene for a disease.

24.4.1 Early Discoveries

Mendel was the first to recognize that there are discrete units of inheritance resulting in different phenotypes as demonstrated by his pea plant experiments. He formulated two laws:

1. **Law of Segregation** – Each individual possesses two alleles for a trait and only one is passed on to its offspring. The offspring will express the phenotype for whichever allele is dominant.

2. **Law of Independent Assortment** – Separate genes for separate traits are passed to offspring independently of one another.

It was later observed that some pairs of phenotypes are not passed on independently which violates the Law of Independent Assortment. This is due to chromosomal linkage where specific genes are generally passed along to offspring in tandem. One would think this would not be possible due to recombination. Recombination is an event during meiosis where a homologous chromosome pair (containing one strand from the mother and one from the father) exchanges fragments and then splits apart. This process generates new combinations of genes for the offspring. Despite recombination, linkage still occurs and it is typically found with genetic loci that are physically close on a chromosome. Linkage Analysis, described later in the text, makes use of this phenomenon to infer disease pathways.

24.4.2 Human Heterozygosity

The technological advances made in sequencing have provided an abundance of Genomic data that has allowed new insights into Human Genetics to be gained. As a result, one important lesson is that human heterozygosity is relatively limited. Any two unrelated people share about 99.5% of their DNA sequence. It turns out that humans are actually among one of the least diverse organisms. The ways in which variation
does occur happens in one of several flavors. ?? depicts the sources of genetic variation. A Single nucleotide polymorphism (SNPs) is a single point changes in a nucleotide base and these genetic markers are most the commonly used to understand disease and potentially implicated genes. The extent of diversity is even further diminished by the fact that very large sets of common variants are typically shared across populations. This observation led to a follow on project to the Human Genome Project called the International HapMap Project. This project aims to catalogue human genetic variation across diverse groups of people.

### Genetic variation

- Single nucleotide polymorphisms (SNPs)
  - 1 every few hundred bp, mutation rate* \( \approx 10^{-9} \)
  - TGCATTGCCTAGGC
  - TGCATTCCCTAGGC

- Short indels (=insertion/deletion)
  - 1 every few kb, mutation rate v. variable
  - TGCATT----TAGGC
  - TGCATTCCCTAGGC

- Microsatellite (STR) repeat number
  - 1 every few kb, mutation rate \( \leq 10^{-4} \)
  - TGCCTACATCATAAGGC
  - TGCCTACATCAAGGC

- Minisatellites
  - 1 every few kb, mutation rate \( \leq 10^{-4} \)

- Repeated genes
  - rRNA, histones

- Large deletions, duplications, inversions
  - Rare, e.g. Y chromosome

* rates per generation

Figure 24.3: Different sources of genetic variation

The fact that so many common variants are shared across populations yields redundancy and hence efficiency in the data. Knowing one SNP in a haplotype can allow you to guess with a degree of certainty which nucleotides exist at other markers of interest. Examining the non-redundant set of SNPs allows one to significantly minimize computational complexity in gene finding problems. ?? illustrates this concept of redundant information gained from haplotypes.

### 24.4.3 Mendelian vs. Complex Traits

Linkage Analysis works well for diseases and traits that are Mendelian in nature. Mendelian traits are those that follow Mendel’s Laws. A mutation in a gene causes a defect that has pathological consequences. ?? shows examples of single-gene, Mendelian diseases. The mutations are severe and obvious and thus culprit genes are located well with computational methods.

?? shows a plot of the number of genes found for traits that are Human Mendelian traits, complex traits, and Human complex traits versus time. The number of genes implicated in Human Mendelian traits has exploded as time (and technology) have progressed. This is not true for complex traits which cannot be tied back to just one mutation. Often times, there is an involvement of many genes and environmental factors that contribute to complex traits. Gene prediction for complex diseases is a much less straightforward problem to solve.

One example of a complex trait is height. Height is one of the most heritable phenotypes that can be measured. 80-90% of height can be explained by the height of parents and the child’s gender, yet no single gene has been implicated in height. It is postulated that there are 10-100 genes involved in the expression of height.
Due to the nature of complex traits and diseases, it is necessary to perform analysis on whole genomes. Technology has not made acquiring this type of data accurately and at large scales possible until recently. A Genome Wide Association Study, described later in the text, is a method used to infer disease pathways in complex diseases that takes the whole genome into account. The following sections will discuss methods used to infer causal disease pathways through the analysis of genetic data.

## 24.5 Linkage Analysis

Linkage analysis takes advantage of the fact that some genes are passed along to offspring together in order to isolate the location of a gene that causes a disease. If a genetic marker co-segregates with a positive disease status in a family, one can infer that the gene is near (on the same chromosomal arm) as the mutation. This genetic marker may have nothing to do with the disease itself, but it will get you closer to a marker that does cause the disease due to linkage. Illustrates a Linkage Analysis pedigree where red indicates an individual is afflicted with a disease. The goal of the problem is to find a strand of DNA that is common to the afflicted individuals but not common to the rest of the family.

Linkage analysis in its earlier form was carried out by first proposing a possible genetic marker, evaluating the marker against the genome, and then computing what is called the LOD score. Positive LOD scores indicate the presence of linkage while negative LOD score indicate the absence of linkage.
Figure 24.6: Discovery of genes for different disease types versus time.

LOD (logarithm of odds) = \log \left( \frac{\text{Likelihood that the disease is rare}}{\text{Likelihood that the disease is far away}} \right)

The likelihood that the disease was present was calculated using parametric linkage analysis:

\[ L = \sum \sum \cdots \sum \prod_{\text{individual}} P(\phi | g)P(g | \text{parents}) \]

This calculation is a large recursive operation that sums over all possible genotypes for all family members. To simplify the fact that recursion had to evaluate all family members, Elston and Stewart made the simplification with their peeling algorithm that knowing the parents genotypes provided all of the probabilistic information necessary for the children and so this need to recurse over all possible family members was eliminated. Summing over all possibly genotypes is not computationally expensive as long as a single marker is being evaluated. In the case of a single marker, one would be evaluating against the genotypes AA, Aa, and aa. This operation does indeed become more expensive because it is necessary to evaluate against multiple markers since individual markers provide incomplete information about inheritance. Homozygous parents provide no inheritance information about where an allele came from and it becomes necessary to consider multiple genetic markers (haplotypes) along a chromosome to gain a complete picture.
The newest formulation of the Linkage analysis problem lays out two steps. First, an inheritance pattern is inferred from the pedigree data and second, the inheritance information is used to determine that absence or presence of a disease/trait influencing gene. Inheritance vectors are the tools used to capture this data. These vectors are binary:

\[
\begin{align*}
0 & = \text{grand-paternal allele passed on} \\
1 & = \text{grand-maternal allele passed on}
\end{align*}
\]

The inheritance vector captures which founder alleles are carried by each non-founder. ?? demonstrates an example of generating an inheritance vector for a pedigree. Squares indicate males and circles indicate women. Also, the first allele in the pair indicates the paternal allele and the second allele indicates the maternal allele. In the example, vector are generated for the children at the very bottom. For the first child, the paternal allele 'A' came from the grandfather and is denoted with a '0'. The maternal allele 'C' came from the grandmother and is denoted with a '1'. This process is repeated for the second child.

**Inheritance vectors: example 1**

![Figure 24.8: An inheritance vector example](image)

The conversion of genotype information into an inheritance vector allows for different analyses to be performed on the data to predict disease-causing genes. The inheritance vectors can be treated as states in a HMM setup. Recombination causes transitions between states and the possible observations are whether a disease/trait is observed or not. This multipoint analysis is used in GENEHUNTER and other linkage software packages.

### 24.6 Genome Wide Association Studies

Rapid advances in sequencing technology have taken the field from a model of selecting specific regions to be sequenced to sequencing whole genomes rapidly and cost-effectively. For complex diseases, examining the entire genome is important and Genome Wide Association Studies (GWAS) aim to adequately assess the entirety of the data.

There are several key parts to GWAS. First, quality control is very important. Sequencing technologies are not full proof and it’s necessary to account for missing or misinterpreted data. There are metrics that can be used against the data to determine general validity (i.e. Hardy-Weinberg equilibrium). Second, one must test for association. The most straightforward way is to compare frequencies of alleles present for both controls and cases. These tests are performed on the order of $10^5$ to $10^6$ largely independent tests and the statistical significance of each test is evaluated in turn. Third, replication is necessary to fully validate results from a study. There is a huge focus not only on performing GWASs but also on meta-analyzing multiple GWAS. It is possible to pull together various studies with different levels of information due to haplotypes and the fact that large amounts of genetic data are redundant and offer information into what missing data might be. Studies can be combined in a way that fills in the missing pieces of sequenced data so that comparison is possible.
The results of GWAS are typically reported as Manhattan plots and Q-Q plots. Manhattan plots are scatter plots that display a large number of points. For GWAS, the x-axis indicates genome location and the y-axis indicates the negative log of the association p-value for a SNP. This ensures that the smallest p-values, which indicate greatest correlation, are the highest on the chart. This allows the viewer to visually pick out where on the genome a disease seems to be related to the most strongly. Q-Q plots are probability plots for comparing two probability distributions. Q-Q plots are used in the context of GWAS to visually check for confounders in the data. Confounders are variables present in the data other than the disease itself. The most common confounder noted is geographic/ethnic groups. The Q-Q plot plots the expected distribution of test statistics against the x-axis and versus the million SNPs compared to the observed values on the y-axis. Any deviation from Y=X indicates a consistent difference in cases and controls which means that a confounder is present.

The final step in completing a GWAS is to evaluate the biological significance of the finding. It is important to identify the relevant gene, identify the causal DNA variant, discover the molecular function of that variant, and understand how that function impact biological processes involved in disease.

Several lessons have been learned from GWAS so far. First, fewer than \( \frac{1}{3} \) of associations are related to coding or obviously functional variants. Second, 21 of 50 non-coding associations are significantly associated to expression level of a nearby gene. Third, many are associated to regions with no coding gene. Last, the majority are associated to multiple autoimmune or inflammatory diseases. These revelations indicate that there are still many mysteries lurking in the genome waiting to be discovered.

### 24.7 Linkage vs. Association

As an abundance of data allowed researchers to switch from Linkage Analysis to Genome Wide Association Studies, diseases that had been studied in the past under linkage were reevaluated with GWAS. It was commonly found that genes implicated with Linkage Analysis were also implicated with GWAS but not vice versa. It was also commonly found that GWAS implicated many more genes. GWAS are the tool of today for studying the causal pathways that genes affect in diseases. While Linkage Analysis cannot cover both Mendelian and complex diseases, GWAS is able to cover these two areas and offers more complete predictions with the data that is available today.
24.8 Current Research Directions

24.9 Further Reading

24.10 Tools and Techniques

24.11 What Have We Learned?

Dr. Daly demonstrated that the problem of implicating specific genes in disease is very important for personalized medicine. It is not an easy problem to formulate and constantly evolving methods from Linkage Analysis to Genome Wide Association Studies have been applied to datasets that are also rapidly evolving as sequencing technology advances. This has allowed many new insights to be made into the underlying mechanisms of human disease and population variance. Understanding the ways in which populations vary genetically has allowed researchers to devise smarter, more robust methods to tackle these problems. This is a field that is constantly progressing and it will only be a matter of time until the most perplexing of diseases are well understood.

Bibliography
CHAPTER

TWENTYFIVE

MISSING HERETIBILITY

TODO: missing @scribe: add author

List of figures
25.1 Introduction

25.2 Current Research Directions

25.3 Further Reading

25.4 Tools and Techniques

25.5 What Have We Learned?

Bibliography